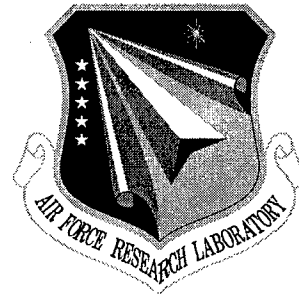AFRL-IF-RS-TR-1999-166
Final Technical Report
August 1999

SCALABLE AND EXTENSIBLE COOPERATIVE INFORMATION SYSTEMS

University of California, Los Angeles

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
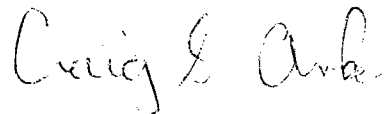ROME RESEARCH SITE
ROME, NEW YORK**

DTIC QUALITY INSPECTED 4

19991015 023

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1999-166 has been reviewed and is approved for publication.

APPROVED: *Craig S. Anken*

CRAIG S. ANKEN
Project Engineer

FOR THE DIRECTOR: *Northrup Fowler*

NORTHRUP FOWLER, III, Technical Advisor
Information Technology Division
Information Directorate

SCALABLE AND EXTENSIBLE COOPERATIVE INFORMATION SYSTEMS

Wesley W. Chu

Contractor: University of California, Los Angeles
Contract Number: F30602-94-C-0207
Effective Date of Contract: 1 August 1994
Contract Expiration Date: 30 December 1998
Program Code Number: 6.2
Short Title of Work: Scalable and Extensible Cooperative
                         Information Systems

Period of Work Covered: Aug 94 - Dec 98

Principal Investigator: Wesley W. Chu
              Phone: (310) 825-2047
AFRL Project Engineer: Craig S. Anken
              Phone: (315) 330-4833

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1999	Final Aug 94 - Dec 98

4. TITLE AND SUBTITLE

SCALABLE AND EXTENSIBLE COOPERATIVE INFORMATION SYSTEMS

5. FUNDING NUMBERS

C - F30602-94-C-0207
PE - 62301E
PR - B397
TA - 00
WU - 01

6. AUTHOR(S)

Wesley W. Chu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of California, Los Angeles
Department of Computer Science
405 Hilgard Avenue
Los Angeles CA 90095

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency   Air Force Research Laboratory/IFTB
3701 North Fairfax Drive   525 Brooks Road
Arlington VA 22203-1714   Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-1999-166

11. SUPPLEMENTARY NOTES

Air Force Research Laboratory Project Engineer: Craig S. Anken/IFTB/4833

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Conventional query answering requires the user to have detailed knowledge of the database schema. As the size and the number of types of databases in the system increase, it becomes impractical for the user to know the entire database schema. This effort developed a cooperative information system that uses a knowledge base to provide approximate matching of different data structures and schemas. Further, based on the user profile and application, the system is capable of query relaxation and modification, and is able to provide approximate answers when an exact query answer is not available. The user can also provide explicit relaxation parameters such as approximate range, near to, similar to, as well as relaxation. A prototype was developed, CoBase, that supports structured data for transportation applications, and have also extended the cooperative methodology to the image domain.

14. SUBJECT TERMS

Cooperative Rules, Cooperative Operators, Rule Relaxation

15. NUMBER OF PAGES

150

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

Executive Summary

Conventional query answering requires the user to have detailed knowledge of the database schema. As the size and the number of types of databases in the system increase, it becomes impractical for the user to know the entire database schema. Under this contract, we developed a cooperative information system that uses a knowledge base to provide approximate matching of different data structures and schemas. Further, based on the user profile and application, the system is capable of query relaxation and modification, and is able to provide approximate answers when an exact query answer is not available. The user can also provide explicit relaxation parameters such as approximate range, near to, similar to, as well as relaxation. We have developed a prototype system, CoBase, that supports structured data for transportation applications, and have also extended the cooperative methodology to the image domain (e.g., GIS and medical images).

The relaxation manager controls the query relaxation process based on the user profile, query context, and query conditions. The relaxation control (e.g., relaxation order, relaxable range, unacceptable list) not only reduces search scope but also provides the user with more precise answers.

We developed a structured way to represent the knowledge called Type Abstraction Hierarchy (TAH). Object types, tuples, and attributes are represented at different abstraction levels in the TAH. Therefore, TAH provides multi-level knowledge representation (from various levels of concept to indexes) and is more general than the conventional object data model representation, which consists of two levels of knowledge representation. The TAH can be generated automatically for structured data (e.g., database), and we have also extended it to image data (e.g., features). A knowledge editor will be developed to enable the user or domain expert to edit or manually create the TAH. Since the knowledge update to the TAH is localized, it eases knowledge acquisition and maintenance. The detailed results are summarized in the paper entitled "An Error-Based Conceptual Clustering Method for Providing Approximate Query Answers," CACM Virtual Extension (http://www/acm.org/pubs/CACM/extension).

Associative query answering provides additional relevant information to the queries that is not explicitly asked, but is of interest to the user. For a given query, associative information may be derived from past user query cases based on the user type and the query context. A case-based reasoning approach that matches query features has been developed. Query feature consists of the query topic, the output attribute list, and the selection constraints. The similarity of the query feature is defined and can be evaluated from the semantic model that is derived from the database schema. The results are summarized in [CZ97].

For the purpose of scalability and reuse, we incorporated the cooperative operations into three mediators: relaxation mediator, association mediator, and explanation mediator. The relaxation mediator provides approximate matching of information from different sources with different granularities, scope, and abstraction. The association mediator provides relevant query information not asked by the user. The explanation mediator is based on the relaxation action traces and describes the relaxation process and the quality of the query answer. These mediators can be distributed, interconnected and reused, thus our mediator architecture is scalable. In our proposed mediator architecture, there are two aspects of scalability: scalability of the cooperative mediators, and scalability of the mediator architecture. Let us first address the scalability of the cooperative mediators.

The relaxation mediator utilizes a knowledge structure—TypeAbstraction Hierarchies (TAHs)—that provide multiple-level knowledge representation. For each database attribute type, its TAH is a classification hierarchy of its possible values. Operations are provided to traverse the hierarchy, such as generalization (up) and specialization (down). Query condition values are relaxed to their semantic neighbors until an approximate answer is produced. Explicit relaxation operations and control of the relaxation process can also be provided by the user. The TAH can be generated automatically from relational databases for both numerical and non-numerical attribute values [CCHY96]. The computation time required to generate the TAH increases approximately in the order of square with the number of tuples. We have also generalized the methodology to generate TAH for spatial and temporal features of the image data. Thus, spatial as well as temporal relaxation can be performed. We are able to process spatial queries based on spatial feature characteristics and interrelationships rather than on pixels, thus reducing the query processing time. Based on application context and the user's constraints and requirements, the relaxation manager reduces the search scope and generates more accurate answers.

Explanations may occur at many points during the relaxation process. The user can select paragraphs, sentences, or words in the explanation for definition, elaboration, justification, etc. The explanation mediator takes the action traces of the cooperative operations, and puts the explanation goals on the trace object.

The size and processing time of the trace scales linearly with the number of actions performed by the mediator (typically 10 to 50 actions per query). The invocation rules determine when explanation events occur. The invocation rules, which are based on the trace object types and user model types, produce the explanation goal. The explanation operators determine the content of explanations. The explanation templates are associated with types, not instances, and hence have good reuse properties. Based on the user models, the mediator generates the appropriate explanation text for the user. The algorithms used in explanation are summarized in [MC99].

Let us now discuss the scalability of the *cooperative mediator architecture*. The cooperative mediator is a software module comprised of the following components:

- *Precondition.* This component specifies the data to be imported into the local mediation process. The condition will be a statement of the data types and qualifications of the necessary data conditions. Specifically, the preconditions are queries to the Intelligent Directory/Dictionary or other mediators specifying the data to be retrieved and the constraints that must be satisfied. To each data input, we attach a weight specifying the input degree of certainty.

- *Postcondition.* This component specifies the information abstraction produced by the mediator. It will be a definition of the abstract data types produced and the constraints that are satisfied by the output information. A mediator's postcondition will specify the certainty of its output abstraction.

- *Mediation.* This component performs the actual abstraction of the mediator. The mediator takes the input specified by the precondition, applies the context-specific knowledge encoded in the mediation component, and produces abstractions characterized by the postcondition.

The cooperative mediators are able to transform lower-level, heterogeneous, distributed data and knowledge into a higher level of abstraction.

Since the mediators have a uniform interface, they can be interconnected. Each mediator can use the output of other mediators as input. Further, mediator protocol are developed for them to communicate and interact with each other for performing such operations as negotiations, process binding, recovery, commitments, etc. Therefore, out mediator architecture has an infrastructure supporting a large number of cooperative mediators for intelligent integration of information sources. For a more detailed discussion, see [CYC96].

Most database interfaces provide poor guidance for ad hoc query formulation which burdens the user to learn or to know precisely the query language and the database schema. An ideal query interface should assume that users may have little technical knowledge and possibly possess no knowledge concerning the schema of the database. Many current and future applications of DBMSs, e.g., scientific computing and decision support, require user interaction based on many ad hoc queries, instead of the conventional innovations of pre-compiled and stored application programs. As database schema becomes larger and more complex, there is a need to develop a high-level query interface to allow users to specify queries by high-level concepts and constraints.

We developed a new approach for query formulation based on a semantic graph model; this provides a semantic representation of the data in the database augmented with user-defined relationships. The graph model can be semi-automatically generated from the database schema. The query formulator allows users to specify their requests and constraints in high-level concepts to formulate queries instead of using a database query language. The query formulator completes a query based on the user input and ranks the formulated query candidates according to the probabilistic information measure. English-like query descriptions can also be provided for the user to resolve ambiguity when multiple queries are formulated from user input. Further, the system allows the user to interact with the system

and select the desired query. A prototype system using the proposed technology with a multimodal interface consisting of GUI and voice interface has been implemented at UCLA. The formulator is operating on top of the cooperative database system (CoBase) to formulate SQL queries. The results are summarized in a report included in the Appendix entitled, "Query Formulation from High-level Concepts with Multi-modal Interface" [ZCKM98].

Technology Transfer

CoBase technology has been successfully applied to the Electronic Warfare (EW) domain for identifying emitter features under noisy conditions (with Lockheed Martin) and to the medical domain for approximate matching of medical images by feature and content (with the UCLA Medical School). A technology transfer was made to the DARPA logistics planning application (GLAD), and an IFD was performed in May 1996 (with BBN). CoBase was also successfully integrated in a DARPA Advanced Logistics Planning (ALP) demo in September 1997. CoBase demonstrated relaxation capabilities for a remote client on data provided by a remote server using a CORBA interface.

Currently, under DARPA support, we developing relaxation plug-in modules for the ALP clustering architecture.

References

Detailed descriptions and results of research successfully conducted under the Scalable and Extensible Cooperative Information Systems contract (F30602-94-C-0207) are contained in the following technical papers. These papers are included as an appendix to this report.

[CCHY96]   Chu, W. W., K. Chiang, C. C. Hsu, and H. Yau, "An Error-Based Conceptual Clustering Method for Providing Approximate Query Answers," in Communications of the ACM, Vol. 39, No. 13, December 1996 (http://www.acm.org/pubs/cacm/extension).

[CYC96]   Chu, W. W., H. Yang, K. Chiang, M. Minock, G. Chow, and C. Larson, "CoBase: A Scalable and Extensible Cooperative Information System," in Journal of Intelligent Information System, Vol. 6, 1996, pp. 223-260.

[CZ97]   Chu, W. W. and G. Zhang, "Associative Query Answering via Query Feature Similarity," Proceedings of the International Conference on Intelligent Information Systems, Grand Bahama Island, the Bahamas, December 8-10, 1997.

[CZ97]   Chu, W. W. and G. Zhang, "Associations and Roles in Object-Oriented Modeling," Proceedings of the 16th International Conference on Conceptual Modeling, Los Angeles, California, November 3-6, 1997.

[MC99]   Minock, M. and W. W. Chu, "Explanation Over Inference Hierarchies in Active Mediation Applications."

[ZCKM98]   Zhang, G., W. W. Chu, G. Kong, and F. Meng, "Query Formulation from High-level Concepts with Multi-modal Interface," October 1998.

Appendix to Final Report
Scalable and Extensible Cooperative information Systems

Contents

An Error-based Conceptual Clustering Method for Providing Approximate Query Answers

Wesley W. Chu, Kuorong Chiang, Chih-Cheng Hsu, Henrick Yau

Knowledge discovered from databases can be used to abstract the data into high level concepts. The discovered concept, or abstraction, makes it easier for users to understand the data and can be used to process queries intelligently [2, 11, 9, 5]. In particular, abstraction can be used to derive approximate answers—when the objects requested by a query are not available, the query conditions can be relaxed to their corresponding abstraction where "neighborhood objects" can be found and returned as the approximate answers.

For clustering objects into "neighborhoods," two methods can be used: statistical clustering and numerical taxonomy [19], or conceptual clustering [15, 13, 14]. In statistical clustering and numerical taxonomy, most similarity metrics are defined between pairs of objects. Objects are pair-wise clustered in a bottom up manner until all objects are in a single cluster. In conceptual clustering, the "goodness measures" are usually defined for the overall *partitioning* of objects instead of for pairs of objects. Clustering methods are designed to maximize the goodness measure. We find the conceptual clustering technique more suitable for approximate query answering [5, 8] since the goodness measure can be directly related to the expected quality of approximate answers. By minimizing the expected error associated with the query relaxation, we are able to derive the *best* clustering of objects.

Most current *conceptual clustering* systems only deal with non-numerical values. That is, the values are categorical and can only be equal or not equal. These systems only consider *frequency distribution* of data because they are concerned with the *grouping* of values rather than the *values* themselves. Such clustering, however, is inadequate for providing approximate answers as illustrated in the following example. Consider two clusters $C_1 = \{0, 100\}$ and $C_2 = \{0, 1\}$. C_1 and C_2 are equivalent based on frequency distribution alone because they both have two distinct values. As approximate answers, however, they are very different because the values in C_2 are much closer to each other than those in C_1. Thus, for defining a neighborhood of objects, C_2 is much better than C_1.

For discovering abstraction, therefore, a clustering method must consider both the frequency and the *value* distributions of data. In this paper, we propose a DIstribution Sensitive Clustering (DISC) method that considers both the frequency and the value distributions in discovering high level concepts from data.

The rest of the paper is organized as follows. After a brief discussion of prior related work, we develop the notion of *relaxation error* as a quality measure for clusters. Then we present the DISC algorithm for generating concept hierarchies, Type Abstraction Hierarchies (TAH), that minimize the relaxation error for a single attribute as well as multiple attributes. Next, we present applications of TAH for providing approximate query answers and for feature-based image retrieval. To show the effectiveness of our discovered knowledge for approximate query answering, we compare the results derived from TAH generated by DISC with those of the traditional index tree. Finally, we address the issue of maintaining TAHs and offer a feasible solution of keeping TAHS up to date.

Related Work

Prior work in discretization aims at decreasing cardinality of data by maximizing/minimizing certain heuristic measures. A commonly used measure is the information entropy [17]. It can be shown that the entropy is maximum when the data is partitioned most evenly. (We call this the ME method [3, 21].) However, no semantic meaning is attached to the resultant clusters because the discretization is concerned with the frequency distribution rather than the value distribution in the cluster. Therefore, the ME method is not suitable for abstracting numerical data.

COBWEB [10], a conceptual clustering system, uses *category utility* (*CU*) as a quality measure to classify the objects described by a set of attributes into a classification tree. Formally, for a partition from a class C to N mutually exclusive classes $C_1,...,C_N$, the category utility (*CU*) is defined as the increase in the average class... *goodness*[1]. That is,

$$CU(C_1,...,C_N) = \frac{\sum_{k=1}^{N} P(C_k)G(C_k) - G(C)}{N} \tag{1}$$

where $P(C_k)$ is the occurrence probability of C_k in C, and $G(C_k)$ and $G(C)$ are the goodness functions for C_k and C, respectively. That is,

$$G(C_k) = \sum_{a \in A} \sum_{x_i^a \in X_k^a} P(x_i^a)^2 \qquad (2)$$

$$G(C) = \sum_{a \in A} \sum_{x_i^a \in X^a} P(x_i^a)^2 \qquad (3)$$

where A is the set of all the attributes, and X_k^a and X^a are the distinct values of attribute a in C_k and C respectively.

COBWEB cannot be used for abstracting numerical data; it only deals with categorical data. Moreover, its classification tree serves as *the database* for the instances and requires a large storage space. Furthermore, matching of objects with existing classes is time consuming. For providing approximate answers, we want to build a classification tree *in addition to* the original database. The tree can be viewed as an index so that its storage and traversal should be very efficient.

CLASSIT [12], an extension of the COBWEB system, classifies numerical attributes where each attribute is normally distributed. Concepts are represented in terms of mean and standard deviation (σ). $1/\sigma$ is used as the quality measure for a concept and the data is classified to maximize the weighted sum of $1/\sigma$. The problem of using CLASSIT for approximate query answering is that data are assumed to be normally distributed. This is not true in many cases; our experience with a transportation database shows many numerical attributes are skewed and multimodal. Further, we also observed multiple "impulse values" which represent occurrences of certain attribute values with a very high frequency. In addition, CLASSIT cannot represent concepts with a single value because $\sigma = 0$, making $1/\sigma = \infty$. To overcome this problem, CLASSIT introduces a system parameter called *acuity* which is the minimum value allowed for σ. *Acuity* avoids the infinite quality problem, but it may cause undesirable data classification for approximate query answering. For example, an impulse value, say v_i, cannot be the only value in a concept; to satisfy the *acuity* constraint, the concept is forced to include other values, say v_j. Thus, whenever v_j is relaxed, it is relaxed to $\{v_i, v_j\}$ which may be a very poor approximation for v_j.

To remedy these shortcomings, we develop a new goodness measure for the classification of numerical data in view of approximate query answering.

Relaxation Error—A Goodness Measure for Clustering Numerical Values

To deal with numerical values, we need to generalize category utility. To simplify our presentation, let us consider the single-attribute case. For a class $C = \{x_1,..., x_n\}$, (3) reduces to

$$G(C) = \sum_{j=1}^{n} P(x_i)^2 \qquad (4)$$

where x_i is the i-th distinct attribute value and $P(x_i)$ is the occurring probability of x_i in C.

The goodness measure can be interpreted as follows. The class C in (4) is represented by an urn of balls, each of them representing an attribute value. The number of balls of a particular color represents the frequency of occurrence of the corresponding attribute value. We randomly draw two balls from the urn, one at a time with replacement. If the two balls have the same value, we score 1. Otherwise, we score 0. If we do this experiment a large number of times, the expected score we have will be the goodness of the cluster $G(C)$. Let $s(x_i, x_j)$ be the score for the drawn values x_i and x_j, then $s(x_i, x_j) = 1$ if $x_i = x_j$ and 0 otherwise. Thus, (4) becomes

$$G(C) = \sum_{j=1}^{n} \sum_{j=1}^{n} P(x_i)P(x_j)s(x_i, x_j) \qquad (5)$$

Equation (5) cannot serve as a quality measure for numerical values as illustrated in the following: consider the two clusters $C_1 = \{0, 100\}$ and $C_2 = \{0, 1\}$ as mentioned in previously. According to (5), C_1 is as good as C_2: $G(C_1) = G(C_2) = 0.5$. Intuitively, this is unsatisfying because the values in C_2 are much closer to each other than those in C_1. As a result, a label such as *small* can be attached to C_2 but not to C_1.

Based on the above observation, a good quality measure for numerical values can be derived by substituting $s(x_i, x_j)$ in (5) with the absolute difference between x_i and x_j. The result is called the *relaxation error* of C, $RE_1(C)$. Formally,

7

$$RE_1(C) = \sum_{j=1}^{n} \sum_{j=1}^{n} P(x_i)P(x_j) \, | x_i - x_j | \qquad (6)$$

That is, $RE_1(C)$ is the average pair-wise difference among values in C. Using $RE_1(C)$ for the above example, we have $RE_1(C) = 50$ and $RE_1(C_2) = 0.5$. Clearly, C_2 is much better than C_1.

$RE_1(C)$ can also be interpreted from the standpoint of query relaxation. Let us define the *relaxation error* of x_i, $RE_1(x_i)$, as the average difference from x_i to x_j, $j = 1,..., n$. Formally,

$$RE_1(x_i) = \sum_{j=1}^{n} P(x_j) \, | x_i - x_j | \qquad (7)$$

where $P(x_j)$ is the occurring probability of x_j in C. $RE_1(x_i)$ can be used to measure the quality of an approximate answer where x_i in a query is relaxed to x_j, $j = 1,..., n$. Summing $RE_1(x_i)$ over all values x_i in C, we have

$$RE_1(C) = \sum_{j=1}^{n} P(x_i)RE_1(x_i). \qquad (8)$$

Thus, $RE_1(C)$ is the expected error of relaxing any value in C.

If $RE_1(C)$ is large, query relaxation based on C may produce very poor approximate answers. To overcome this problem, we can partition C into sub-clusters to reduce relaxation error. Given a partition $P = \{C_1, C_2, ..., C_N\}$ of C, the *relaxation error of the partition P* is defined as

$$RE_1(P) = \sum_{k=1}^{N} P(C_k)RE_1(C_k) \qquad (9)$$

where $P(C_k)$ equals the number of tuples in C_k devided by the number of tuples in C. In general, $RE_1(P) < RE_1(C)$.

Using relaxation error, the category utility can be defined as the relaxation error reduction per sub-cluster, that is,

$$CU = \frac{\sum_{k=1}^{N} P(C_k)[1 - RE_1(C_k)] - [1 - RE_1(C)]}{N} = \frac{RE_1(C) - \sum_{k=1}^{N} P(C_k)RE_1(C_k)}{N} \qquad (10)$$

Relaxation Error for Multiple Attributes

As mentioned above, relaxation error is the expected pair-wise difference between values in a cluster. To extend the notion of relaxation error from a single attribute to multiple attributes, we shall consider distance between tuples instead of difference between values. Given two m-attribute tuples $t_i = \{x_1,..., x_m\}$ and $t_j = \{y_1,..., y_m\}$, their distance is defined as

$$D(t_i, t_j) = \sum_{k=1}^{m} W_k \frac{| x_k - y_k |}{\Delta_k} \qquad (11)$$

where W_k and Δ_k are the weight and the normalization constant for the k-th attribute, respectively. W_k can be assigned to reflect relative importance among attributes. Δ_k is necessary for summing up differences from different attributes because different attributes have different distributions. For example, consider two attributes from the *AIRCRAFT* relation (Table 5): Runway-Width and Max-Weight. Max-Weight has a much greater value range, so its value differences tend to be much greater than that of Runway-Width and will dominate $D(t_i, t_j)$. Thus, normalization is necessary for summing up pair-wise differences from different attributes.

There are two possible normalization factors. One is $x_{max} - x_{min}$, and another is $RE_1(X)$, the relaxation error of the attribute X. Both factors are based on pair-wise difference: $x_{max} - x_{min}$ is the *maximum* pair-wise difference and $RE_1(X)$ is the *average* pair-wise difference. $x_{max} - x_{min}$ is much easier to compute but is easily subjected to influence by out-liers of values. $RE_1(X)$ is more costly to compute but is not easily subjected to influence from out-liers. Since the cost of computing $RE_1(X)$ is small (linear with respect to the number of distinct values in X), we select $RE_1(X)$ as the normalization factor.

To see the effect of the normalization factor, consider the attributes Runway-Width and Max-Weight again. The relaxation error for Runway-Width and Max-Weight are 12.89 and 265,602.38, respectively. Using $RE_1(X)$ as the normalization factor, a difference of 12.89 in Runway-Width is as significant as a differene of 265,602.38 in

Max-Weight.

Given a cluster of n m-attribute tuples $C = \{t_1,..., t_n\}$, the relaxation error for C is defined as the average pair-wise distance among tuples in C, that is,

$$RE(C) = \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{n} P(t_i)P(t_j)D(t_i, t_j) \tag{12}$$

where $P(t_i)$ and $P(t_j)$ are the probabilities of tuples t_i and t_j, respectively. The division by m in (12) normalizes $RE(C)$ per attribute and allows us to compare relaxation errors computed from different numbers of attributes. The category utility (CU) for multiple attributes can be obtained by simply substituting $RE_1(C)$ in Eq (10) by its multi-attribute counterpart $RE(C)$ in Eq (12).

The Clustering Algorithm

We shall now present a class of DISC algorithms for clustering numerical values. We shall present the algorithm for a single attribute, and then extend it for multiple attributes.

The Clustering Algorithm for a Single Attribute

Given a cluster with n distinct values, the number of partitions is exponential with respect to n, so the best partition according to (10) takes exponential time to find. To reduce computation complexity, we shall only consider binary partitions (i.e., $N = 2$ in (10)). Later we shall show a simple hill climbing strategy can be used for obtaining N-ary partitions from binary partitions.

Our method is top down: we start from one cluster consisting of all the values of an attribute, and then we find "cuts"[2] to recursively partition the cluster into smaller clusters. The partition result is a concept hierarchy called Type Abstraction Hierarchy (TAH). The clustering algorithm is called the DISC (DIstribution Sensitive Clustering) Method and is given in Table 1.

In [7], an implementation of the algorithm BinaryCut is presented whose time complexity is $O(n)$. Since DISC needs to execute BinaryCut at most $n - 1$ times to generate a TAH, the worst case time complexity of DISC is $O(n^2)$. (The average case time complexity of DISC is $O(n \log n)$.)

N-ary Partitioning

N-ary partitions can be obtained from binary partitions by a hill climbing method. Starting from a binary partition, the sub-cluster with greater relaxation error is selected for further cutting. We shall use CU as a measure to determine if the newly formed partition is better than the previous one. If the CU of the binary partition is greater than that of the tri-nary partition, then the tri-nary partition is dropped and the cutting is terminated. Otherwise, the tri-nary partition is selected and the cutting process continues until it reaches the point where a cut decreases CU. The

Algorithm DISC(C)
    if the number of distinct values $\in C < T$, return /* T is a threshold */
    let cut = the best cut returned by **BinaryCut(C)**
    partition values in C based on cut
    let the resultant sub-clusters be C_1 and C_2
        call DISC(C_1) and DISC(C_2)
Algorithm BinaryCut(C)
/* input cluster $C = \{x_1,..., x_n\}$ */
    for $h = 1$ to $n - 1$ /* evaluate each cut */
        Let P be the partition with clusters $C_1 = \{x_1,..., x_h\}$ and $C_2\{x_{h+1},..., x_n\}$
        compute CU
        if $CU > maxCU$ then
            $max\ CU = CU$, $cut = h$ /* the best cut */
    Return cut as the best cut

Table 1. The algorithms DISC and BinaryCut

Algorithm *N*-ary Partition(C)
let C_1 and C_2 be the two sub-clusters of C
compute CU for the partition C_1, C_2
for $N = 2$ to $n - 1$
        let C_i be the sub-cluster of C with maximum relaxation error call
        Binary Cut to find the best sub-clusters C_{i1} and C_{i2} of C_i
        compute and store CU for the partition $C_1, . . ., C_{i-1}, C_{i1}, C_{i2}, C_{i+1}, ..., C_N$
        if current CU is less than the previous CU
            stop
      else
            replace C_i by C_{i1} and C_{i2}
/* the result is an *N*-ary partition of C */

Table 2. The *N*-ary partition algorithm

procedure is outlined in Table 2. Note that *N*-ary TAH can be generated by replacing BinaryCut with *N*-ary Partition in DISC algorithm.

The Clustering Algorithm for Multiple Attributes

Query relaxation for multiple attributes using multiple single-attribute TAHs relaxes each attribute independently disregarding the relationships that might exist among attributes. This may not be adequate for the applications where attributes are dependent.[3] In addition, using multiple TAHs is inefficient since it may need many iterations of query modification and database access before approximate answers are found. Furthermore, relaxation control for multiple TAHs is more complex since there are a large number of possible orders for relaxing attributes. In general, we can only rely on simple heuristics such as *best first* or *minimal coverage first* [7] to guide the relaxation. These heuristics cannot guarantee best approximate answers since they are rules of thumb and are not necessarily accurate.

    Most of the above mentioned difficulties can be overcome by using Multi-attribute TAH (MTAH) for the relaxation of multiple attributes. Although MTAHs can be generated from any set of attributes, MTAHs should be generated only from *semantically dependent* attributes. Using MTAHs for query relaxation, interrelated attributes can always be relaxed together. Approximate answers can be derived efficiently using MTAH because only a single query modification and database access is necessary. The approximate answers derived by using MTAH are better than those derived by using multiple TAHs.[4]

    To cluster objects of multiple attributes, DISC can be extended to M-DISC (shown in Table 3). The generated multi-dimensional TAHs are called MTAHs. The algorithm DISC is a special case of M-DISC, and TAH is a special case of MTAH.

    Let us now consider the time complexity of M-DISC. Let m be the number of attributes and n be the number of distinct attribute values. The computation of relaxation error for a single attribute takes $O(n \log n)$ to complete [7]. Since the computation of CU involves computation of relaxation error for m attributes, its complexity is $O(mn \log n)$. The nested loop in M-DISC is executed mn times, so the time complexity of M-DISC is $O(m^2n^2 \log n)$. To generate an MTAH, it takes no more than n calls of M-DISC, therefore, the worst case time complexi-

Algorithm M-DISC(C)
    if the number of objects in $C < T$, return /* T is a threshold */
    for each attribute $a = 1$ to m
        for each possible binary cut h
            compute CU for h
            if $CU > MaxCU$ then /* remember the best cut */
                $MaxCU = CU, BestAttribute = a, cut = h$
    partition C based on cut of the attribute $BestAttribute$
    let the resultant sub-clusters be C_1 and C_2
        call M-DISC(C_1) and M-DISC(C_2)

Table 3. The multi-attribute DISC (M-DISC) algorithm

Algorithm gM-DISC(C)
    if the number of objects in $C < T$, return /* T is a threshold */
    for each attribute $a = 1$ to m
        for each possible binary cut h
            compute ΔRE_a for h
            if $\Delta RE_a > Max\Delta RE$ then /* remember the best cut */
                $Max\Delta RE = \Delta RE_a$, $BestAttribute = a$, $cut = h$
    partition C based on cut of the attribute $BestAttribute$
    let the resultant sub-clusters be C_1 and C_2
        call gM-DISC(C_1) and gM-DISC(C_2)

Table 4. The greedy M-DISC algorithm: gM-DISC

ty of generating an MTAH is $O(m^2 n^3 \log n)$. The average case time complexity is $O(m^2 n^2 (\log n)^2)$ since M-DISC needs only to be called $\log n$ times on the average.

gM-DISC: a Greedy M-DISC Algorithm

The time complexity of M-DISC can be greatly reduced if we are willing to accept an approximation. We shall present the heuristic for obtaining the sub-optimal cuts and the resulted *greedy* M-DISC algorithm here. Later, we shall show that the performance of the greedy M-DISC is comparable with that of the non-greedy M-DISC.

For ease of discussion, we shall say a partition is *disjointed* on an attribute a if the values of a are grouped into non-overlapping ranges. In general, a partition will have only one disjointed attribute which among all attributes will have the largest reduction of relaxation error. Thus, the magnitude of CU is generally dominated by the reduction of relaxation error on the disjointed attribute. (Recall that the meaning of CU is the relaxation error reduction due to the partition.) Therefore, we may replace CU with the relaxation error reduction on the disjointed attribute without losing too much accuracy.

Let ΔRE_a be the reduction of relaxation error for attribute a. Replacing CU with ΔRE_a in M-DISC, we have the greedy M-DISC (shown in Table 4).

The relaxation error reduction ΔRE_a used in gM-DISC is normalized such that ΔRE_a from different attributes can be compared. The worst case time complexity for generating an MTAH using gM-DISC is $O(mn^2)$ where
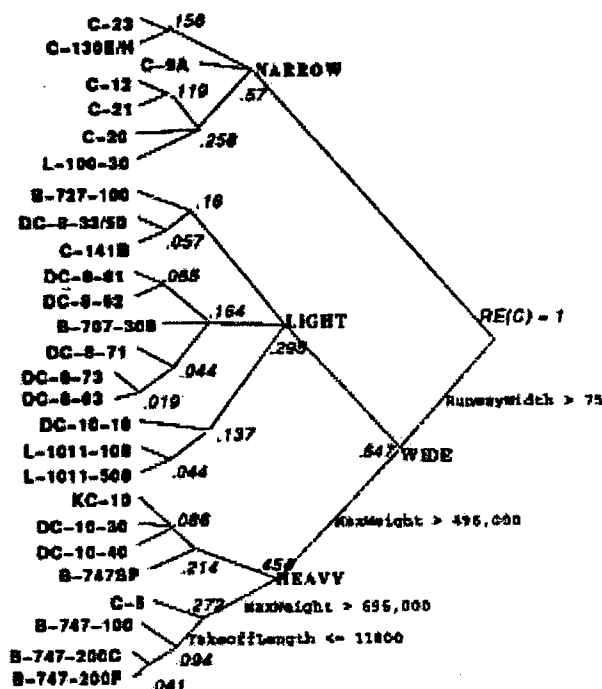

Figure 1. The MTAH for *AIRCRAFT*

Aircraft Type	Length Takeoff	Length Landing	Runway Width	Weight Max.	Weight Empty	Storage Space
C-5	13600	5000	90	769000	665000	21508
C-141B	9000	5000	90	343000	239000	10454
KC-10	11800	5400	90	590000	414000	11600
C-130E	6250	3000	60	173700	NULL	5173
C-130H	6250	3000	60	173700	NULL	5173
C-9A	8100	3200	10	108000	90000	4325
C-12	4500	4500	75	12500	11000	915
C-21	6000	3400	75	18300	13500	747
C-20	5000	5000	75	69700	44000	2518
C-23	4000	3000	60	22900	16200	1690
B-707-300	10800	7000	90	336600	230000	5600
DC-8-33	9400	5500	90	315000	224000	10000
DC-8-50	9400	5500	90	315000	224000	10000
DC-8-61	10400	6100	90	325000	261000	8100
DC-8-62	11500	6100	90	350000	230000	8825
DC-8-63	10400	6100	90	355000	261000	10800
DC-8-71	8850	6100	90	355000	261000	10800
DC-8-73	9800	6100	90	355000	261000	10800
DC-10-10	9500	5800	90	440000	335000	11000
DC-10-30	11850	6100	90	572000	391000	11600
DC-10-40	10600	5800	90	572000	391000	11600
B-747SP	7500	5600	90	696000	410000	14100
B-747-100	9500	6600	90	750000	526400	17500
B-747-200C	11800	6600	90	775000	590000	17500
B-747-200F	10500	6600	90	775000	590000	17500
L-1011-100	10640	7300	90	466000	320000	10800
L-1011-500	9760	7300	90	496000	338000	10800
B-727-100	9200	4800	90	207500	160000	5600
L-100-30	6000	4900	75	155000	122000	5800

Table 5. The database table AIRCRAFT

m is the number of attributes and n is the number of distinct values. The average case time complexity is $O(mn \log n)$.

An Example

We shall present an example here using the *AIRCRAFT* relation shown in Table 5. We use M-DISC to generate an MTAH based on the six numerical attributes. Each attribute is assumed to be equally important in characterizing an aircraft. In certain cases, however, this may not be true and different *weights* can be assigned to attributes for a better classification of objects. Using equal weights, the resulting MTAH is shown in Figure 1.

One advantage of MTAH is that it is easy to understand. Each node of the MTAH stores conditions that all instances covered by the node must satisfy. In addition, each node stores the cuts information which shows *how* the objects are partitioned. For example, consider the three top level clusters. The aircrafts C-23,..., L-100-30 have *narrow* runway (Runway-Width $\leq$ 75), while the other aircrafts have *wide* (Runway-Width $>$ 75). The *wide* aircrafts are further classified by Max-Weight: the aircrafts L-727-100,..., L-1011-500 are *light* (Max-Weight $\leq$ 496,000) while the aircrafts KC-10,..., B-747-200F are *heavy* (Max-Weight $>$ 496,000). The rest of the MTAH can be understood in the same manner.

Note that the relaxation errors (normalized by RE(C)) are computed (from Eq 12) and shown at each node. Note that the relaxation errors of the MTAH are monotonically decreasing from the root to the leaves, indicating the quality of nodes increases towards the leaves.

Applications

Application to Approximate Query Answering

We shall now use an example to show how the TAHs generated by DISC can be used for providing approximate answers.

Example. Consider the query "find a cargo with size 300 square feet and weight 740 kg."
The corresponding SQL query is

select CARGO-ID

from CARGOS

where SQUARE-FEET = 300 and WEIGHT = 740.

The query conditions are too specific; no answers are found for the query. To obtain approximate answers, the query is modified by relaxing cargo size and weight according to the TAHs for SQUARE-FEET and WEIGHT (see Figure 2). (These TAHs are generated by DISC from a transportation database for the CARGOS relation.) As a result, the query is modified to

select CARGO-ID

from CARGOS

where $294 \leq$ SQUARE-FEET ≤ 300 and $737 \leq$ WEIGHT ≤ 741.

The modified query is submitted to the database and the following cargo is returned:

CARGO-ID	SQUARE-FEET	WEIGHT
10	296	740

The quality of the answer can be measured by its relaxation error from equation (12). From the CARGOS relation, we obtain Δ(SQUARE-FEET) = 11.95 and Δ(WEIGHT) = 9.88. Thus, assuming the two attributes SQUARE-FEET and WEIGHT are equally important, the relaxation error is 0.168. This quality measure corresponds to a 16.8% relaxation. A 100% relaxation corresponds to no query conditions on SQUARE-FEET and WEIGHT.
If more answers are needed, the conditions can be further relaxed by moving one more step up the TAHs:

select CARGO-ID

from CARGOS

where $294 \leq$ SQUARE-FEET ≤ 306 and $737 \leq$ WEIGHT ≤ 749.

The following four cargos are returned for this query:

CARGO-ID	SQUARE-FEET	WEIGHT
10	296	740
21	301	737
30	304	746
44	306	745

For the above answers, the relaxation error is 0.334, which is greater than the previous one because the values deviate more from the specified ones in this case.
Thus, from the above example, we see that the TAHs generated by DISC can be used for relaxing query conditions to obtain "neighborhood answers" as well as their quality measures.

Application to Feature-based Image Retrieval

Images can be retrieved based on features and contents [4, 16]. Extracting the essential features that can capture
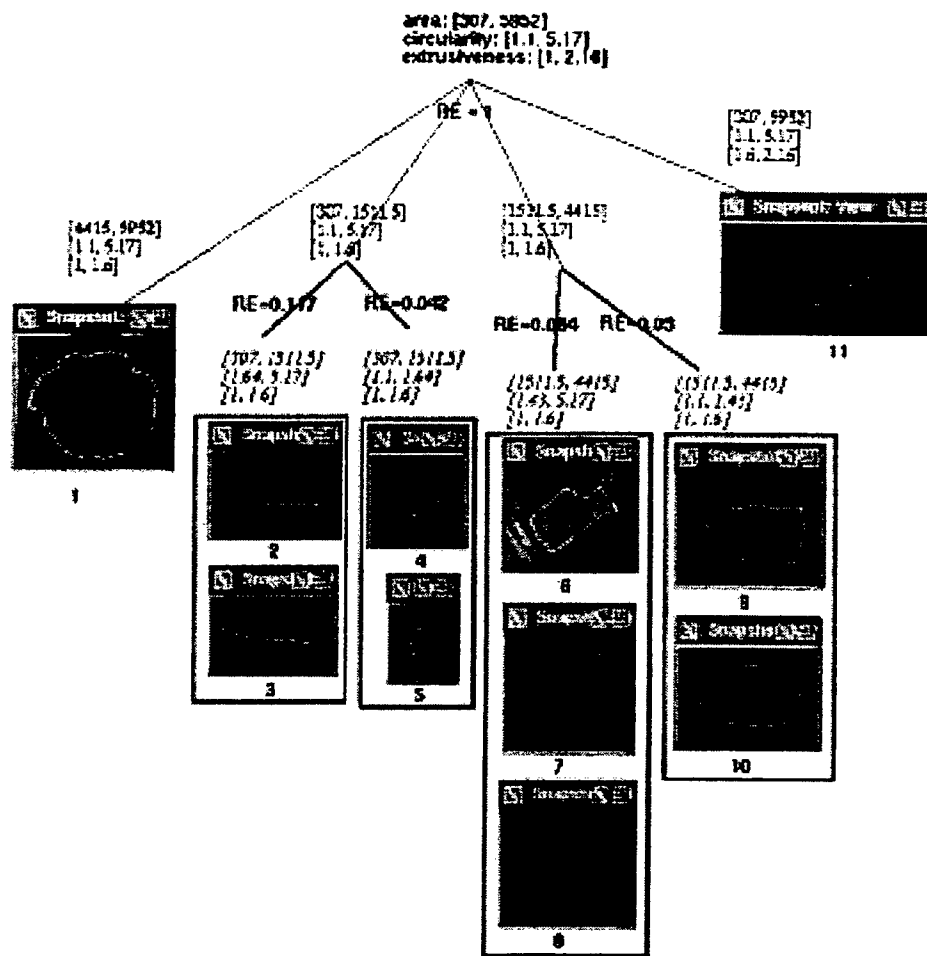
Figure 3. An MTAH for 11 lung tumor contours generated by M-DISC based on *area, circularity,* and *extrusiveness.*
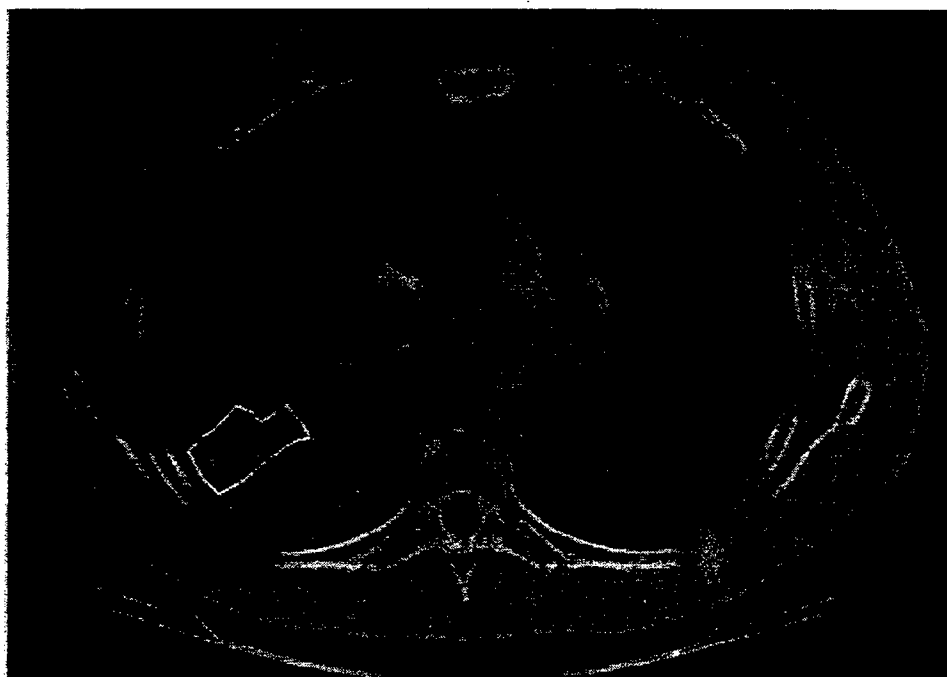

Figure 4. The CT scanned lung image for image 6 (in Figure 3) with the lung tumor contour outlined.

Appendix to Final Report
Scalable and Extensible Cooperative information Systems

Contents

An Error-based Conceptual Clustering Method for Providing Approximate Query Answers

Wesley W. Chu, Kuorong Chiang, Chih-Cheng Hsu, Henrick Yau

Knowledge discovered from databases can be used to abstract the data into high level concepts. The discovered concept, or abstraction, makes it easier for users to understand the data and can be used to process queries intelligently [2, 11, 9, 5]. In particular, abstraction can be used to derive approximate answers—when the objects requested by a query are not available, the query conditions can be relaxed to their corresponding abstraction where "neighborhood objects" can be found and returned as the approximate answers.

For clustering objects into "neighborhoods," two methods can be used: statistical clustering and numerical taxonomy [19], or conceptual clustering [15, 13, 14]. In statistical clustering and numerical taxonomy, most similarity metrics are defined between pairs of objects. Objects are pair-wise clustered in a bottom up manner until all objects are in a single cluster. In conceptual clustering, the "goodness measures" are usually defined for the overall *partitioning* of objects instead of for pairs of objects. Clustering methods are designed to maximize the goodness measure. We find the conceptual clustering technique more suitable for approximate query answering [5, 8] since the goodness measure can be directly related to the expected quality of approximate answers. By minimizing the expected error associated with the query relaxation, we are able to derive the *best* clustering of objects.

Most current *conceptual clustering* systems only deal with non-numerical values. That is, the values are categorical and can only be equal or not equal. These systems only consider *frequency distribution* of data because they are concerned with the *grouping* of values rather than the *values* themselves. Such clustering, however, is inadequate for providing approximate answers as illustrated in the following example. Consider two clusters $C_1 = \{0, 100\}$ and $C_2 = \{0, 1\}$. C_1 and C_2 are equivalent based on frequency distribution alone because they both have two distinct values. As approximate answers, however, they are very different because the values in C_2 are much closer to each other than those in C_1. Thus, for defining a neighborhood of objects, C_2 is much better than C_1.

For discovering abstraction, therefore, a clustering method must consider both the frequency and the *value* distributions of data. In this paper, we propose a DIstribution Sensitive Clustering (DISC) method that considers both the frequency and the value distributions in discovering high level concepts from data.

The rest of the paper is organized as follows. After a brief discussion of prior related work, we develop the notion of *relaxation error* as a quality measure for clusters. Then we present the DISC algorithm for generating concept hierarchies, Type Abstraction Hierarchies (TAH), that minimize the relaxation error for a single attribute as well as multiple attributes. Next, we present applications of TAH for providing approximate query answers and for feature-based image retrieval. To show the effectiveness of our discovered knowledge for approximate query answering, we compare the results derived from TAH generated by DISC with those of the traditional index tree. Finally, we address the issue of maintaining TAHs and offer a feasible solution of keeping TAHS up to date.

Related Work

Prior work in discretization aims at decreasing cardinality of data by maximizing/minimizing certain heuristic measures. A commonly used measure is the information entropy [17]. It can be shown that the entropy is maximum when the data is partitioned most evenly. (We call this the ME method [3, 21].) However, no semantic meaning is attached to the resultant clusters because the discretization is concerned with the frequency distribution rather than the value distribution in the cluster. Therefore, the ME method is not suitable for abstracting numerical data.

COBWEB [10], a conceptual clustering system, uses *category utility* (CU) as a quality measure to classify the objects described by a set of attributes into a classification tree. Formally, for a partition from a class C to N mutually exclusive classes $C_1,...,C_N$, the category utility (CU) is defined as the increase in the average class... goodness[1]. That is,

$$CU(C_1,...,C_N) = \frac{\sum_{k=1}^{N} P(C_k)G(C_k) - G(C)}{N} \quad (1)$$

where $P(C_k)$ is the occurrence probability of C_k in C, and $G(C_k)$ and $G(C)$ are the goodness functions for C_k and C, respectively. That is,

$$G(C_k) = \sum_{a \in A} \sum_{x_i^q \in X_k^q} P(x_i^q)^2 \qquad (2)$$

$$G(C) = \sum_{a \in A} \sum_{x_i^q \in X^a} P(x_i^q)^2 \qquad (3)$$

where A is the set of all the attributes, and X_k^q and X^a are the distinct values of attribute a in C_k and C respectively.

COBWEB cannot be used for abstracting numerical data; it only deals with categorical data. Moreover, its classification tree serves as *the database* for the instances and requires a large storage space. Furthermore, matching of objects with existing classes is time consuming. For providing approximate answers, we want to build a classification tree *in addition to* the original database. The tree can be viewed as an index so that its storage and traversal should be very efficient.

CLASSIT [12], an extension of the COBWEB system, classifies numerical attributes where each attribute is normally distributed. Concepts are represented in terms of mean and standard deviation (σ). $1/\sigma$ is used as the quality measure for a concept and the data is classified to maximize the weighted sum of $1/\sigma$. The problem of using CLASSIT for approximate query answering is that data are assumed to be normally distributed. This is not true in many cases; our experience with a transportation database shows many numerical attributes are skewed and multi-modal. Further, we also observed multiple "impulse values" which represent occurrences of certain attribute values with a very high frequency. In addition, CLASSIT cannot represent concepts with a single value because $\sigma = 0$, making $1/\sigma = \infty$. To overcome this problem, CLASSIT introduces a system parameter called *acuity* which is the minimum value allowed for σ. *Acuity* avoids the infinite quality problem, but it may cause undesirable data classification for approximate query answering. For example, an impulse value, say v_i, cannot be the only value in a concept; to satisfy the *acuity* constraint, the concept is forced to include other values, say v_j. Thus, whenever v_j is relaxed, it is relaxed to $\{v_i, v_j\}$ which may be a very poor approximation for v_j.

To remedy these shortcomings, we develop a new goodness measure for the classification of numerical data in view of approximate query answering.

Relaxation Error—A Goodness Measure for Clustering Numerical Values

To deal with numerical values, we need to generalize category utility. To simplify our presentation, let us consider the single-attribute case. For a class $C = \{x_1, ..., x_n\}$, (3) reduces to

$$G(C) = \sum_{j=1}^{n} P(x_i)^2 \qquad (4)$$

where x_i is the i-th distinct attribute value and $P(x_i)$ is the occurring probability of x_i in C.

The goodness measure can be interpreted as follows. The class C in (4) is represented by an urn of balls, each of them representing an attribute value. The number of balls of a particular color represents the frequency of occurrence of the corresponding attribute value. We randomly draw two balls from the urn, one at a time with replacement. If the two balls have the same value, we score 1. Otherwise, we score 0. If we do this experiment a large number of times, the expected score we have will be the goodness of the cluster $G(C)$. Let $s(x_i, x_j)$ be the score for the drawn values x_i and x_j, then $s(x_i, x_j) = 1$ if $x_i = x_j$ and 0 otherwise. Thus, (4) becomes

$$G(C) = \sum_{j=1}^{n} \sum_{j=1}^{n} P(x_i)P(x_j)s(x_i, x_j) \qquad (5)$$

Equation (5) cannot serve as a quality measure for numerical values as illustrated in the following: consider the two clusters $C_1 = \{0, 100\}$ and $C_2 = \{0, 1\}$ as mentioned in previously. According to (5), C_1 is as good as C_2: $G(C_1) = G(C_2) = 0.5$. Intuitively, this is unsatisfying because the values in C_2 are much closer to each other than those in C_1. As a result, a label such as *small* can be attached to C_2 but not to C_1.

Based on the above observation, a good quality measure for numerical values can be derived by substituting $s(x_i, x_j)$ in (5) with the absolute difference between x_i and x_j. The result is called the *relaxation error* of C, $RE_1(C)$. Formally,

$$RE_1(C) = \sum_{j=1}^{n} \sum_{j=1}^{n} P(x_i)P(x_j) \, |x_i - x_j| \qquad (6)$$

That is, $RE_1(C)$ is the average pair-wise difference among values in C. Using $RE_1(C)$ for the above example, we have $RE_1(C) = 50$ and $RE_1(C_2) = 0.5$. Clearly, C_2 is much better than C_1.

$RE_1(C)$ can also be interpreted from the standpoint of query relaxation. Let us define the *relaxation error* of x_i, $RE_1(x_i)$, as the average difference from x_i to x_j, $j = 1,..., n$. Formally,

$$RE_1(x_i) = \sum_{j=1}^{n} P(x_j) \, |x_i - x_j| \qquad (7)$$

where $P(x_j)$ is the occurring probability of x_j in C. $RE_1(x_i)$ can be used to measure the quality of an approximate answer where x_i in a query is relaxed to x_j, $j = 1,..., n$. Summing $RE_1(x_i)$ over all values x_i in C, we have

$$RE_1(C) = \sum_{j=1}^{n} P(x_i)RE_1(x_i). \qquad (8)$$

Thus, $RE_1(C)$ is the expected error of relaxing any value in C.

If $RE_1(C)$ is large, query relaxation based on C may produce very poor approximate answers. To overcome this problem, we can partition C into sub-clusters to reduce relaxation error. Given a partition $P = \{C_1, C_2, ..., C_N\}$ of C, the *relaxation error of the partition P* is defined as

$$RE_1(P) = \sum_{k=1}^{N} P(C_k)RE_1(C_k) \qquad (9)$$

where $P(C_k)$ equals the number of tuples in C_k devided by the number of tuples in C. In general, $RE_1(P) < RE_1(C)$.

Using relaxation error, the category utility can be defined as the relaxation error reduction per sub-cluster, that is,

$$CU = \frac{\sum_{k=1}^{N} P(C_k)[1 - RE_1(C_k)] - [1 - RE_1(C)]}{N} = \frac{RE_1(C) - \sum_{k=1}^{N} P(C_k)RE_1(C_k)}{N} \qquad (10)$$

Relaxation Error for Multiple Attributes

As mentioned above, relaxation error is the expected pair-wise difference between values in a cluster. To extend the notion of relaxation error from a single attribute to multiple attributes, we shall consider distance between tuples instead of difference between values. Given two m-attribute tuples $t_i = \{x_1,..., x_m\}$ and $t_j = \{y_1,..., y_m\}$, their distance is defined as

$$D(t_i, t_j) = \sum_{k=1}^{m} W_k \frac{|x_k - y_k|}{\Delta_k} \qquad (11)$$

where W_k and Δ_k are the weight and the normalization constant for the k-th attribute, respectively. W_k can be assigned to reflect relative importance among attributes. Δ_k is necessary for summing up differences from different attributes because different attributes have different distributions. For example, consider two attributes from the *AIRCRAFT* relation (Table 5): Runway-Width and Max-Weight. Max-Weight has a much greater value range, so its value differences tend to be much greater than that of Runway-Width and will dominate $D(t_i, t_j)$. Thus, normalization is necessary for summing up pair-wise differences from different attributes.

There are two possible normalization factors. One is $x_{max} - x_{min}$, and another is $RE_1(X)$, the relaxation error of the attribute X. Both factors are based on pair-wise difference: $x_{max} - x_{min}$ is the *maximum* pair-wise difference and $RE_1(X)$ is the *average* pair-wise difference. $x_{max} - x_{min}$ is much easier to compute but is easily subjected to influence by out-liers of values. $RE_1(X)$ is more costly to compute but is not easily subjected to influence from out-liers. Since the cost of computing $RE_1(X)$ is small (linear with respect to the number of distinct values in X), we select $RE_1(X)$ as the normalization factor.

To see the effect of the normalization factor, consider the attributes Runway-Width and Max-Weight again. The relaxation error for Runway-Width and Max-Weight are 12.89 and 265,602.38, respectively. Using $RE_1(X)$ as the normalization factor, a difference of 12.89 in Runway-Width is as significant as a differene of 265,602.38 in

Max-Weight.

Given a cluster of n m-attribute tuples $C = \{t_1,..., t_n\}$, the relaxation error for C is defined as the average pair-wise distance among tuples in C, that is,

$$RE(C) = \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{n}P(t_i)P(t_j)D(t_i, t_j) \qquad (12)$$

where $P(t_i)$ and $P(t_j)$ are the probabilities of tuples t_i and t_j, respectively. The division by m in (12) normalizes $RE(C)$ per attribute and allows us to compare relaxation errors computed from different numbers of attributes. The category utility (CU) for multiple attributes can be obtained by simply substituting $RE_1(C)$ in Eq (10) by its multi-attribute counterpart $RE(C)$ in Eq (12).

The Clustering Algorithm

We shall now present a class of DISC algorithms for clustering numerical values. We shall present the algorithm for a single attribute, and then extend it for multiple attributes.

The Clustering Algorithm for a Single Attribute

Given a cluster with n distinct values, the number of partitions is exponential with respect to n, so the best partition according to (10) takes exponential time to find. To reduce computation complexity, we shall only consider binary partitions (i.e., $N = 2$ in (10)). Later we shall show a simple hill climbing strategy can be used for obtaining N-ary partitions from binary partitions.

Our method is top down: we start from one cluster consisting of all the values of an attribute, and then we find "cuts"[2] to recursively partition the cluster into smaller clusters. The partition result is a concept hierarchy called Type Abstraction Hierarchy (TAH). The clustering algorithm is called the DISC (DIstribution Sensitive Clustering) Method and is given in Table 1.

In [7], an implementation of the algorithm BinaryCut is presented whose time complexity is $O(n)$. Since DISC needs to execute BinaryCut at most $n - 1$ times to generate a TAH, the worst case time complexity of DISC is $O(n^2)$. (The average case time complexity of DISC is $O(n \log n)$.)

N-ary Partitioning

N-ary partitions can be obtained from binary partitions by a hill climbing method. Starting from a binary partition, the sub-cluster with greater relaxation error is selected for further cutting. We shall use CU as a measure to determine if the newly formed partition is better than the previous one. If the CU of the binary partition is greater than that of the tri-nary partition, then the tri-nary partition is dropped and the cutting is terminated. Otherwise, the tri-nary partition is selected and the cutting process continues until it reaches the point where a cut decreases CU. The

Algorithm DISC(C)
   if the number of distinct values $\in C < T$, return /* T is a threshold */
   let cut = the best cut returned by **BinaryCut(C)**
   partition values in C based on cut
   let the resultant sub-clusters be C_1 and C_2
      call DISC(C_1) and DISC(C_2)
Algorithm BinaryCut(C)
/* input cluster $C = \{x_1,..., x_n\}$ */
   for $h = 1$ to $n - 1$ /* evaluate each cut */
      Let P be the partition with clusters $C_1 = \{x_1,..., x_h\}$ and $C_2\{x_{h+1},..., x_n\}$
      compute CU
      if $CU > maxCU$ then
         $max\ CU = CU$, $cut = h$ /* the best cut */
   Return cut as the best cut

Table 1. The algorithms DISC and BinaryCut

Algorithm _N_-ary Partition(C)

let C_1 and C_2 be the two sub-clusters of C

compute CU for the partition C_1, C_2

for $N = 2$ to $n - 1$

      let C_i be the sub-cluster of C with maximum relaxation error call

      Binary Cut to find the best sub-clusters C_{i1} and C_{i2} of C_i

      compute and store CU for the partition $C_1, \ldots, C_{i-1}, C_{i1}, C_{i2}, C_{i+1}, \ldots, C_N$

      if current CU is less than the previous CU

            stop

      else

            replace C_i by C_{i1} and C_{i2}

/* the result is an _N_-ary partition of C */

Table 2. The _N_-ary partition algorithm

procedure is outlined in Table 2. Note that _N_-ary TAH can be generated by replacing BinaryCut with _N_-ary Partition in DISC algorithm.

The Clustering Algorithm for Multiple Attributes

Query relaxation for multiple attributes using multiple single-attribute TAHs relaxes each attribute independently disregarding the relationships that might exist among attributes. This may not be adequate for the applications where attributes are dependent.[3] In addition, using multiple TAHs is inefficient since it may need many iterations of query modification and database access before approximate answers are found. Furthermore, relaxation control for multiple TAHs is more complex since there are a large number of possible orders for relaxing attributes. In general, we can only rely on simple heuristics such as _best first_ or _minimal coverage first_ [7] to guide the relaxation. These heuristics cannot guarantee best approximate answers since they are rules of thumb and are not necessarily accurate.

Most of the above mentioned difficulties can be overcome by using Multi-attribute TAH (MTAH) for the relaxation of multiple attributes. Although MTAHs can be generated from any set of attributes, MTAHs should be generated only from _semantically dependent_ attributes. Using MTAHs for query relaxation, interrelated attributes can always be relaxed together. Approximate answers can be derived efficiently using MTAH because only a single query modification and database access is necessary. The approximate answers derived by using MTAH are better than those derived by using multiple TAHs.[4]

To cluster objects of multiple attributes, DISC can be extended to M-DISC (shown in Table 3). The generated multi-dimensional TAHs are called MTAHs. The algorithm DISC is a special case of M-DISC, and TAH is a special case of MTAH.

Let us now consider the time complexity of M-DISC. Let m be the number of attributes and n be the number of distinct attribute values. The computation of relaxation error for a single attribute takes $O(n \log n)$ to complete [7]. Since the computation of CU involves computation of relaxation error for m attributes, its complexity is $O(mn \log n)$. The nested loop in M-DISC is executed mn times, so the time complexity of M-DISC is $O(m^2n^2 \log n)$. To generate an MTAH, it takes no more than n calls of M-DISC, therefore, the worst case time complexi-

Algorithm M-DISC(C)

    if the number of objects in $C < T$, return /* T is a threshold */

    for each attribute $a = 1$ to m

        for each possible binary cut h

            compute CU for h

            if $CU > MaxCU$ then /* remember the best cut */

                $MaxCU = CU$, _BestAttribute_ $= a$, _cut_ $= h$

    partition C based on _cut_ of the attribute _BestAttribute_

    let the resultant sub-clusters be C_1 and C_2

        call M-DISC(C_1) and M-DISC(C_2)

Table 3. The multi-attribute DISC (M-DISC) algorithm

TAH for SQUARE-FEET   TAH for WEIGHT
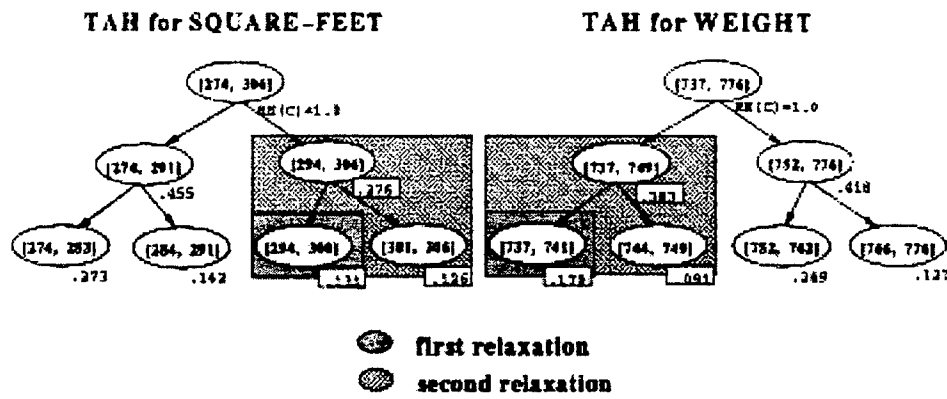

⊛   first relaxation

⊛   second relaxation

Figure 2. TAHs for SQUARE-FEET and WEIGHT generated by DISC from a transportation database. Higher nodes have higher relaxation errors.

TUMOR ID	AREA (*pixel²*)	CIRCULARITY	EXTRUSIVENESS
1	5952.5	1.15	1.03
2	819.5	2.17	1.02
3	743.5	2.8	1.04
4	699	1.1	1
5	307	1.12	1
6	1753.5	1.52	1.07
7	2203.5	1.59	1.04
8	2748.5	1.59	1.05
9	2754	1.27	1
10	2877.5	1.18	1
11	1478.5	5.17	2.16

Table 6. Measurement of features for eleven lung tumors

the specific aspects of the image object depends on context and image domains. Consider lung tumors as an example, three features can be used to classify the shape of lung tumors: *area*, *circularity*, and *extrusiveness*, where *circularity* is defined as $\frac{perimeter^2}{4 \times \pi \times area}$, and *extrusiveness* is defined as $\frac{area\_of\_convex\_hull\_of\_contour}{area}$[20] (Based on this set of features, we can use M-DISC to classify the images and generate an MTAH, as shown in Figure 3.

We can retrieve images with similar feature characteristics from the MTAH. The MTAH leaf nodes represent tumor contours obtained from CT scanned chest images. For example, contour 6 (in Figure 3) is taken from the image in Figure 4. Using this MTAH, contours 7 and 8 are returned as the similar images of contour 6.

For contours with more complex shapes, additional features may be needed to capture the shape characteristics. One approach is based on a prior shape knowledge to decompose the complex shape into several simple ones [18]. We can represent these decomposed simple shapes by an object-oriented shape model which allows us to selectively combine these simpler shapes for retrieving a specific image.

In many current image database systems, such as VIMS [1], similar images are retrieved based on mean and standard deviations of each extracted feature. As a result, similarity is analyzed only based on each attribute separately. Since these features may be dependent, such methods represent a shortcoming. In our approach, however, all the related features can be jointly considered, and thus, it yields more accurate feature clustering.

Performance Comparison of DISC with ME

Single Attribute

Empirical results based on a large transportation database show that clusters discovered by DISC have less relaxation error than those by the Maximum Entropy method (ME). It can be shown [6] that only when the data distribution is symmetrical at the median will the ME method and DISC perform equally well. For skewed distributions, which hold for most data in the transportation database, DISC performs better than ME. Empirical results show

	MTAH	gMTAH	ME-Tree	E-S
accuracy	0.85	0.84	0.68	1.0
efficiency	0.54	0.53	0.64	0.011
relaxation error	1.14	1.17	1.57	1.0

Table 7. Performance comparison of the MTAH, gMTAH, ME tree, and the exhaustive search

that the performance improvement of DISC over ME increases as the skewness increases.

Multiple Attributes

In addition to the relaxation error, we shall introduce two additional performance measures, *accuracy* of the answer and *efficiency* of the retrieval[5], to compare the performance of DISC with ME:

$$\text{accuracy of the answer} = \frac{\text{retrieved relevant answers}}{\text{all relevant answers}}$$

$$\text{efficiency of the retrieval} = \frac{\text{retrieved relevant answers}}{\text{all retrieved answers}}$$

where "all relevant answers" are the best k answers determined by exhaustive search. In general, there is a trade-off between the two measures: the higher the accuracy of the answer, the lower the efficiency of the retrieval.

We generate two MTAHs (one from the algorithm M-DISC and another from the greedy gM-DISC) and an ME tree[6] based on attributes Longitudes and Latitudes of 972 geographical locations from a transportation database. We generate 500 queries with the form: "find the k locations nearest to the target (*long, lat*)" where k is randomly selected from 1 to 20, and *long* and *lat* are randomly generated based on the distributions from the location relation in the database. To answer the query using the MTAHs or the ME tree, we first locate the most specific node in the tree that "covers" (*long, lat*) and contains k' locations where $k' \geq k$. Then we compute the k' distances to (*long, lat*) and use them to select the nearest k locations. The accuracy, efficiency, and the relaxation error from the best k answers to the target location (*long, lat*), averaged over the 500 queries, are shown in Table 7. For easy comparison, performance is given relative to that of the exhaustive search (E-S). We label the MTAH generated by the greedy algorithm gM-DISC as gMTAH.

Notice that the answers provided by the MTAHs (i.e., MTAH and gMTAH) are more accurate (i.e., closer to the target location) than those by the ME tree. Further, the MTAHs are more efficient than the exhaustive search, yet provide answers close to those generated by the exhaustive search with errors less than 17%. We also note that MTAH and gMTAH yield comparable performances. This confirms that the heuristics used in the greedy algorithm gM-DISC provides good approximation.

Maintenance of TAHs

Since the quality of TAH affects the quality of derived approximate answers, TAHs should be kept up to date. One simple way for maintaining TAHs is to regenerate them whenever an update occurs. This approach is not desirable, however, because it causes a lot of overhead for the database system. Although each update changes the distribution of data (thus changing the quality of the corresponding TAHs), it by itself may not be significant enough to warrant a TAH regeneration. TAH regeneration is only necessary when the cumulative effect of updates has greatly degraded the TAHs.

The quality of a TAH can be monitored by comparing the derived approximate answers to the expected relaxation error which is computed at TAH generation time and recorded at each node of the TAH. For example, consider the applications query mentioned earlier. The relaxation error of the first approximate answer is 0.168. From the TAHs in Figure 2, the expected quality of this approximate answer is 0.131 for SQUARE-FEET and 0.170 for WEIGHT, with an average quality of (0.131 + 0.170)/2 = 0.151. Comparison between the actual quality (0.168) and the expected quality (0.151) of the node shows the difference is small, and therefore, the quality of the TAHs is still good and need not be updated.

When the derived approximate answers significantly deviate from the expected quality, then the quality of the TAH is deemed to be inadequate and a regeneration is necessary. The following incremental TAH regeneration procedure can be used. First, identify the node within the TAH that has the worst query relaxations. Apply partial

TAH regeneration for all the database instances covered by the node. After several such partial regenerations, we then initiate a complete TAH regeneration.

Conclusion

In this paper, we present a Multi-attribute DIstribution Sensitive Clustering method (M-DISC) for numerical attribute values. For m attribute of n distinct values, M-DISC generates a Multi-attribute Type Abstraction Hierarchy (MTAH) in $O(m^2 n^2 (\log n)^2)$ time. gM-DISC, a greedy version of M-DISC, yields comparable result with that of M-DISC but only takes $O(mn \log n)$ to generate an MTAH. Thus, gM-DISC is more suitable for tables with a large number of tuples.

For clustering m dependent attributes, our experience have shown that the relaxation error from using MTAH is smaller than from m single-attribute TAHs because single-attribute TAHs cannot take the attribute dependency relationship into consideration. Further, using MTAH is more efficient than using multiple single-attribute TAHs in processing queries because MTAH requires fewer relaxation steps than multiple single-attribute TAHs, thus reducing the number of database access.

We have used M-DISC to generate MTAHs for a large transportation database which consists of 94 relations, the largest one of which has 12 attributes and 195,598 tuples. M-DISC generates approximately 400 numerical TAHs and MTAHs in less than a few hours of processing time on a Sun Sparc 10 Workstation.

The generated TAHs and MTAHs are used in the *Cooperative Database System* (CoBase) [8] at UCLA for providing approximate answers to structured data and for approximate matching of image features. The approximate query answers derived from MTAHs are empirically evaluated in terms of accuracy, efficiency, and relaxation error. The results reveal that the approximate query answers derived from the MTAH generated by DISC are better than those derived from an index tree generated by the Maximum Entropy method.

Appendix. Comparison of Using MTAH to Multiple Single-Attribute TAHs for Query Relaxation On Multiple Numerical Attributes

We shall use a simplified model to illustrate how the dependence relationships which are captured by MTAHs, but not captured by single-attribute TAHs, is useful in guiding the relaxation process.

Considering data with two attributes, we can represent it as a 2-dimensional problem space with tuples represented as points distributed randomly on it. A single point P is chosen randomly as the query condition. The area of consideration is restricted to P and the two closest (the notion for distance will be defined below) points to P, denoted as A and B. The problem is to consider the relative effects of using an MTAH versus using 2 single-attribute TAHs to relax the query conditions on the 2 dimensions in certain relaxation order. The goal is to obtain the closest approximate answers.

Without loss of generality, consider A as the closest point to P. There are 3 possible results of relaxation. The answers resulting from traversing the MTAH and single-attribute TAHs may cover {A, P}, {B, P}, or {A, B, P} (see Figures 5 and 6). Since A is the closest point to P, {A, P} is considered to be the *good* answer, {B, P} is considered to be the *poor* answer, and {A, B, P} is considered to be the *average* answer.
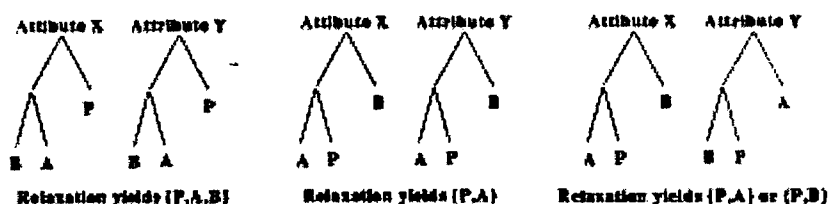

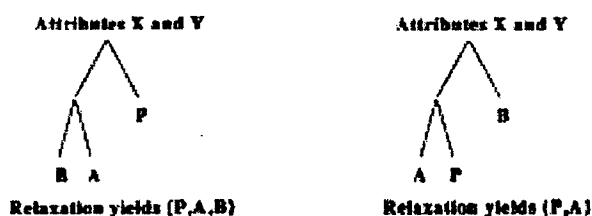Figure 5. Relaxation using multiple single-attribute TAHs

Figure 6. Relaxation using an MTAH

23

Analysis

Distance Measure

The distance between any two points $R(X_R, Y_R)$ and $S(X_S, Y_S)$ is

$$D(R, S) = (|X_R - X_S|)/RE_x + (|Y_R - Y_S|)/RE_y \tag{13}$$

Since points are assumed to be randomly distributed, $RE_x = RE_y$. Thus

$$d(R, S) = |X_R - X_S| + |Y_R - Y_S| \tag{14}$$

which reduces to the Manhattan distance between the points.

Lemma 1 *Closest Distance for Single TAHs.* Given three points in a one-dimensional space, the DISC cutting algorithm always clusters the two closest points together.

   Proof: In a single-dimensional case, $d(R, S) = |X_R - X_S|$. For three points A,B,C projected on a line, there are two possible cuts (AB, BC). DISC cuts on the longer of AB and BC to minimize relaxation error.

Lemma 2 *Closest Distance for MTAHs.* Given three points in a two-dimensional space, the M-DISC cutting algorithm will always cluster the two closest points together.

   Proof: For three points in a two-dimensional space, MTAH clusters two of the points together, minimizing the relaxation error of the cluster. From (12), relaxation error of the cluster is:

$$RE(R, S) = d(R, S)/4$$

Since M-DISC minimizes RE(R, S), d(R,S) is minimized.

   Consider P as the query point and divide the problem space into 4 quadrants, with P at the center. There are 3 possible topologies, depending on the relative positions of A and B, as shown in Figure 5:

Case 1: A and B are in opposite quadrants

In this case, the shortest Manhattan distance is PA. Applying Lemma 2, M-DISC will cluster A, P together. Thus, using MTAH to guide relaxation always yields good results.

   For single-attribute TAHs, there are three possibilities:

(a) $|X_P - X_A| < |X_P - X_B|$ and $|Y_P - Y_A| < |Y_P - Y_B|$

(b) $|X_P - X_A| < |X_P - X_B|$ and $|Y_P - Y_A| > |Y_P - Y_B|$

(c) $|X_P - X_A| > |X_P - X_B|$ and $|Y_P - Y_A| < |Y_P - Y_B|$

   Applying Lemma 1, it is easy to determine that in case (a), DISC will cluster {A, P} together in both dimensions. This yields good relaxation. In cases (b) and (c), DISC will cluster {A, P} together in one dimension and {B, P} together in the other. Using the Minimum Coverage rule[7], for any distribution, relaxation may cover either A or B and the decision is random. As a result, considering all 3 cases, it yields good relaxation in 75% of the time and yields poor relaxation in 25% of the time, while MTAH always yields good relaxation.
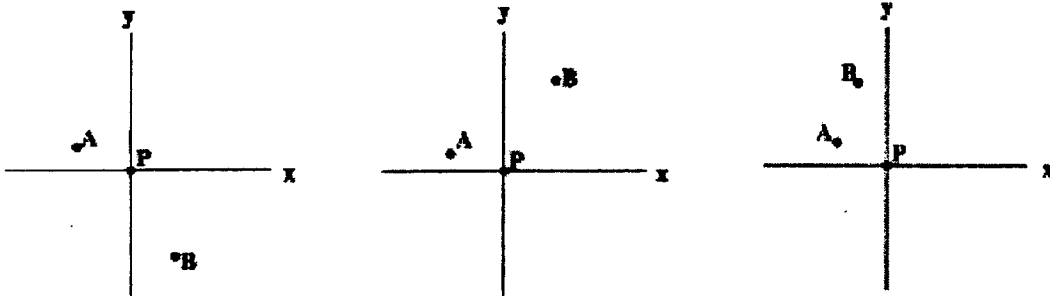
case 1: A,B in opposite quadrants   case 2: A,B in adjacent quadrants   case 3: A,B in the same quadrant

Figure 7. Topologies of 3 points: P, A, and B

Case 2: A and B are in adjacent quadrants
If $d(A, P) \leq d(A, B)$, then relaxation using MTAH yields good results. If $d(A, P) > d(A, B)$, then relaxation using MTAH yields average results.

      In case of single-attribute TAHs, there are some data distribution where DISC would cluster {A, P} together in one attribute and {B, P} together in the other. As in Case 1, relaxation guided by single-attribute TAHs returns poor answers. Relaxation using MTAH is superior since MTAH will always include the closest answer while single-attribute TAHs may miss it.

Case 3: A and B are in the same quadrant
As in Case 2, we can show that relaxation using MTAHs would be good or average, while single-attribute TAHs could miss the good answer.

      The above analysis shows that using an MTAH offers better relaxation on multiple numerical attributes than using multiple single-attribute TAHs. MTAHs successfully retrieve the best answer in all cases, while single-attribute TAHs fail to retrieve the best answer in some cases. This is due to the fact that M-DISC considers information from multiple attributes at the same time.

References

1. Bach J.R., Paul S., and Jain R. A visual information management system for the interactive retrieval of faces. *IEEE Transaction on Knowledge and Data Engineering,* August 1993.
2. Cai T., Cercone N., and Han J. Attribute-oriented induction in relational databases. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases,* pages 213–238. AAAI Press/The MIT Press, 1991.
3. Chin D.K.Y., Wong A.K.C., and Cheung B. Information discovery through hierarchical maximum entropy discretization and synthesis. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases.* AAAI Press/The MIT Press, 1991.
4. Chu W.W., Cardenas A.F., and Taira R.K. KMeD: A knowledge-based multimedia medical distributed database system. *Information System,* Volume 20, No. 2, pages 75–96, 1995.
5. Chu W.W. and Chen Q. Neighborhood and associative query answering. *Journal of Intelligent Information Systems,* 1(3/4), 1992.
6. Chu W.W. and Chiang K. A distribution sensitive clustering method for numerical values. Technical Report 93-0006, UCLA Computer Science Department, 1993.
7. Chu W.W. and Chiang K. Abstraction of high level concepts from numerical values in databases. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases,* July 1994.
8. Chu W.W., Merzbacher M.A., and Berkovich L. The design and implementation of cobase. In *Proceedings of ACM SIGMOD,* Washington, D.C., USA, May 1993.
9. Cuppens F. and Demoloube R. Cooperative answering: a methodology to provide intelligent access to databases. In *Proceedings of the 2nd International Conference on Expert Database Systems,* Virginia, USA, 1988.
10. Fisher D.H. Knowledge acquisition via incremental conceptual clustering. *Machine Learning,* 2(2):139–172, 1987.
11. Gassterland T., Godfrey P., and Minker J. An overview of cooperative answering. *Journal of Intelligent Information Systems,* 1:123–157, 1992.
12. Gennari J., Langley P., and Fisher D. Models of incremental concept formation. *Artificial Intelligence,* 40:11–62, 1989.
13. Hanson S.J. and Bauer M. Conceptual clustering, categorization, and polymorphy. *Machine Learning,* 3:343–372, 1989.
14. Lebowitz M. Experiments with incremental conceptual formation. *Machine Learning,* 2(2):103–138, 1987.
15. Michalski R.S. and Stepp R.E. Learning from observation: Conceptual clustering. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning,* volume 1. Margan Kaufmann Publishers, Inc., 1983.
16. Niblack W., Barber R., Equitz W., Flickner M., Glasman E., Petkovic D., Yanker P., Faloutsos C., and Taubin G. The qbic project: Querying images by content using color, texture, and shape. In *Storage and Retrieval for Images and Video Databases, SPIE,* volume 1908, 1993.
17. Shannon C.E. and Weaver W. *The Mathematical Theory of Communication.* University of Illinois Press,

Urbana, Ill, 1964.

18. Shapiro L. A structural model of shape. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, March 1980.
19. Sneath P.H.A. and Sokal R.R. *Numerical Taxonomy: The Principles and Practice of Numerical Classification.* W.H. Freeman and Company, San Francisco, 1973.
20. Sonka M., Hlavac V., and Boyle R. *Image Processing, Analysis and Machine Vision.* Chapman & Hall Computing, 1993.
21. Wong A.K.C. and Chiu D.K.Y. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):796–805, 1987.

WESLEY W. CHU (wwc@cs.ucla.edu) served as a chair of the UCLA Computer Science Department from 1989 to 1991, and is currently a professor in the department. His research interests include distributed processing, distributed databases, knowledge-based systems.

KUORONG CHIANG (kc8@elsegundoca.ncr.com) is a consulting engineer at NCR working on the optimizer for the Teradata database.

CHIH-CHENG HSU (ccheng@cs.ucla.edu) is a Ph.D. candidate in the Computer Science Department at the University of California, Los Angeles.

HENRICK YAU (hyau@osius.com) is a database consultant specializing in Oracle Financials implementation.

[1]We introduce the term *goodness* for ease of presenting the quality of approximate query answers.

[2]A cut c is a value that separates a cluster of numbers $\{x \mid a \leq x \leq b\}$ into two sub-clusters $\{x \mid a \leq x \leq c\}$ and $\{x \mid c < x \leq b\}$.

[3]Dependency here means that all the attributes as a whole define a coherent concept. For example, the length and width of a rectangle are said to be "semantically" dependent. This kind of dependency should be distinguished from the *functional dependency* in database theory.

[4]A more detailed comparison between MTAH and multiple TAHs is given in the Appendix.

[5]These measures are known as *recall* and *precision* in Information Retrieval.

[6]Since the classification tree generated by the ME method is balanced, it can be viewed as an efficient index.

[7]The rule of Minimum Coverage is used as the relaxation policy for multiple single-attribute TAHs. This rule states that from N different TAHs, select the TAH that yields fewest number of answers so that it results in finer relaxation granularity.

CoBase: A Scalable and Extensible Cooperative Information System*

WESLEY W. CHU, HUA YANG, KUORONG CHIANG,

MICHAEL MINOCK, GLADYS CHOW, CHRIS LARSON

{wwc, hua}@cs.ucla.edu

Computer Science Department, University of California, Los Angeles, CA 90095

Abstract. A new generation of information systems that integrates knowledge base technology with database systems is presented for providing cooperative (approximate, conceptual, and associative) query answering. Based on the database schema and application characteristics, data are organized into Type Abstraction Hierarchies (TAHs). The higher levels of the hierarchy provide a more abstract data representation than the lower levels. Generalization (moving up in the hierarchy), specialization (moving down the hierarchy), and association (moving between hierarchies) are the three key operations in deriving cooperative query answers for the user. Based on the context, the TAHs can be constructed automatically from databases. An intelligent dictionary/directory in the system lists the location and characteristics (*e.g.*, context and user type) of the TAHs. CoBase also has a relaxation manager to provide control for query relaxations. In addition, an explanation system is included to describe the relaxation and association processes and to provide the quality of the relaxed answers. CoBase uses a mediator architecture to provide scalability and extensibility. Each cooperative module, such as relaxation, association, explanation, and TAH management, is implemented as a mediator. Further, an intelligent directory mediator is provided to direct mediator requests to the appropriate service mediators. Mediators communicate with each other via KQML. The GUI includes a map server which allows users to specify queries graphically and incrementally on the map, greatly improving querying capabilities. CoBase has been demonstrated to answer imprecise queries for transportation and logistic planning applications. Currently, we are applying the CoBase methodology to match medical image (X-ray, MRI) features and approximate matching of emitter signals in electronic warfare applications.

Keywords: Approximate query answering, query relaxation, associative query answering, explanation system, mediator architecture, type abstraction hierarchy, conceptual clustering

1. Introduction

Consider asking a query of a human expert. If the posed query has no answer or the complete data for an answer is not available, you do not simply get a null response. The human expert attempts to understand the gist of your query, to suggest or answer related questions, to infer an answer from data that is accessible, or to give an approximate answer. The goal of cooperative database research is to create information systems with these characteristics (Gaasterland et al., 1992). The key is the integration of a knowledge base which represents the data semantics.

In conventional databases, if required data is missing, if an exact answer is unavailable, or if a query is not well-formed with respect to the schema, the database just returns a null answer or an error. An intelligent system would be much more resourceful and cooperative,

---

permitting conceptual level queries (containing concepts that may not be expressed in the database schema) when the user does not know the precise schema, providing approximate answers when some data is missing, or even volunteering associative (relevant) information to the query answer. Cuppens and Demolombe (Cuppens and Demolombe, 1988) provide cooperative answers by rewriting the query to add variables to the query vector, which carry relevant information to the user. The rewrite is not a generalization of the query, it is an extension of the query to provide more information. Motro (Motro, 1988) proposes allowing the user to select the direction of relaxation and thus to indicate which relaxed answers may be of interest. Hemerly, *et al.* (Hemerly et al., 1994) uses a predefined user model and maintains a log of previous interactions to avoid misconstruction when providing additional information. All the above approaches are rule-based and difficult to scale up. To remedy this shortcoming, we will present a structured approach and its implementation to cooperative query answering for database systems.

In this paper, we shall first present the concept of Type Abstraction Hierarchy (TAH) (Chu et al., 1991, Chu and Chiang, 1994) to provide a structured approach for query modification. Methodologies for automatic TAH generation are discussed. Next, we present the cooperative SQL primitives and selected examples. Then, we present the scalable cooperative information system: the relaxation controls for providing efficient query processing and filtering out unsuitable answers for the user, a TAH mediator for providing management and editing of the TAHs in the system, the case-based approach for providing association information to query answers, an explanation mediator, and the inter-mediator communications. Implementation issues are also discussed. The performance of CoBase from a set of sample queries generated from the testbed is reported. Finally, we discuss the technology transfer of CoBase to transportation, medical image databases, and electronic warfare applications.

2. Structured Approach to Cooperative Query Answering

2.1. Query Relaxation via Type Abstraction Hierarchies

Cooperative query answering relaxes a query scope to enlarge the search range or relaxes an answer scope to include additional information. Enlarging and shrinking a query scope can be accomplished by viewing the queried objects at different conceptual levels, since an object representation has wider coverage at a higher level and inversely, more narrow coverage at a lower level. Although linking different level object representations can be made in terms of explicit rules (Cuppens and Demolombe, 1988), such linking lacks a systematic organization to guide the query transformation process. To remedy this problem, we propose the notion of a type abstraction hierarchy (TAH) (Chu et al., 1991, Chu and Chen, 1994) for providing an efficient and organized framework for cooperative query processing. A TAH represents objects at different levels of abstraction. For example, in Figure 1, the *Medium-Range* (*i.e.*, from 4,000 to 8,000 ft.) in the TAH for *runway-length* is a more abstract representation than a specific runway length in the same TAH (*e.g.*, 6,000 ft). Likewise, *SW Tunisia* is a more abstract representation than individual airports (*e.g.*, *Gafsa*). A higher-level and more abstract object representation corresponds to multiple lower-levels and more

specialized object representations. Querying an abstractly represented object is equivalent to querying multiple specialized objects.

A query can be modified by relaxing the query conditions via such operations as *generalization* (moving up the TAH) and *specialization* (moving down the TAH), e.g., from 6000ft to Medium-Range to [4000ft, 8000ft]. In addition, queries may have *conceptual* conditions such as "runway-length = Medium-Range." This condition can be transformed into *specific* query conditions by specialization. Query modification may also be specified explicitly by the user through a set of *cooperative operators* such as 'similar-to', 'approximate', 'near-to', *etc*. This approach was also adopted in the cooperative deductive databases by Gaasterland, *et al.* (Gaasterland et al., 1992) for providing query relaxations.
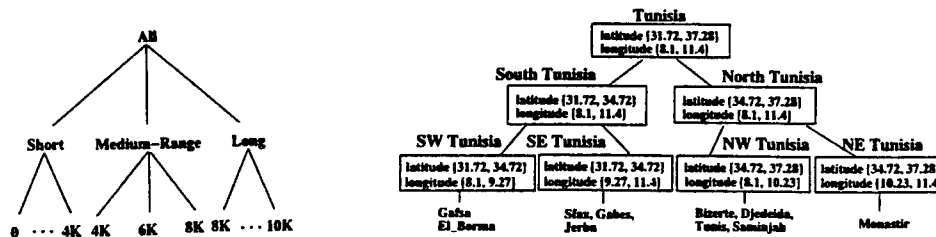

Figure 1. Type Abstraction Hierarchies

The notion of multi-level object representation is not captured by the conventional semantic network and object-oriented database approaches for the following reasons: grouping objects into a class and grouping several classes into a super-class only provide a common "title" (type) for the involved objects without concern for the object instance values and without introducing abstract object representations. Grouping several objects together and identifying their aggregation as a single (complex) object does not provide abstract instance representations for its component objects. Therefore, an object-oriented database deals with information only at two general layers: the meta-layer and the instance layer. Since forming an object-oriented type hierarchy does not introduce new instance values, it is impossible to introduce an additional instance layer. In the Type Abstraction Hierarchy, instances of a super-type and a sub-type may have different representations and can be viewed at different instance layers. Such multiple layer knowledge representation is essential for cooperative query answering.

Knowledge for query relaxation can be expressed as a set of logical rules, but such a rule-based approach (Cuppens and Demolombe, 1989, Hemerly et al., 1994) lacks a systematic organization to guide the query transformation process. TAHs provides a much simpler and more intuitive representation for query relaxation, and do not have the complexity of the inference that exists in the rule-based system. As a result, the TAH structure can easily support flexible relaxation control (see Section 5.2.1), which is important to improve relaxation accuracy and efficiency. Further, knowledge represented in a TAH is customized, thus changes in one TAH represent only a localized update and do not affect other TAHs,

simplifying TAH maintenance (see Section 3.3). We have developed tools to generate TAHs automatically from data sources (see Section 3), which enable our system to scale up and extend to large data sources.

2.2. Properties of the Type Abstraction Hierarchy

Type Abstraction Hierarchy (TAH) is a representation for the abstraction of individual types (atomic or tuple) at instance levels, which is different from the meta-level class/type hierarchy in object-oriented systems.

In our notion of type abstraction, a type can be atomic T (for single attribute), or tuple $T : (T_1, \ldots, T_m)$ (for multiple attributes), where $T_1, \ldots, T_m$ are atomic types. Let us first define the concept of type abstraction.

Type T' is an abstraction of type T, denoted as $T \preceq T'$, means the following:

1. For atomic types T and T', each value in $dom(T')$ represents single or multiple values in $dom(T)$, where $dom(T)$ denotes the domain of type T and $dom(T')$ is the domain of type T'.

2. For tuple types $T : (T_1, \ldots, T_m)$ and $T' : (T'_1, \ldots, T'_m)$, $\forall i \in \{1, \cdots, m\}, T_i \preceq T'_i$.

Typically, the abstraction of a numerical type is a range (or interval) type while the abstraction of a nonnumerical type is a type of conceptual names (*e.g.*, *SW Tunisia*), each representing a set of values (*e.g.*, *Gafsa, El Borma*). We can also assign conceptual names to ranges (*e.g.*, *long runway*).

A TAH is a tree structure constructed from a set of values of a *base type*, which is the most specific type. All other types are abstraction of the base type or other abstract types. The values of the base type are leaf nodes of the TAH. Conceptually, all leaf nodes have the same depth in the TAH[1]. The values of the abstract types constitute the non-leaf nodes of the TAH.

More formally, we denote the base type as T and all abstract types in the TAH as $T', T'', \ldots, T^{(k)}$, where $T^{(k)}$ is the most general type. We have

$$T \preceq T' \preceq T'' \preceq \ldots \preceq T^{(k)}$$

For convenience of notation, $T, T', T'', \ldots$ can also be re-written as $T^{(0)}, T^{(1)}, T^{(2)}, \ldots$, respectively. Values of type $T^{(i)}$ are at i levels above the leaf nodes for $i = 1, \ldots, k$. If we construct a TAH based on a single attribute, then the attribute is the base type. If we construct a TAH based on multiple attributes chosen from one or more relations, then these attributes constitute the base tuple type.

Intuitively, moving up a TAH maps a value of type $T^{(i)}$ to a value of more abstract type $T^{(i+1)}$, which is more general; while moving down the TAH maps from a value of more abstract type $T^{(i+1)}$ to a set of values of type $T^{(i)}$, which is more specific, where $i = 0, \ldots, k - 1$. Relaxation for a value (of the base type) is to map it to a more abstract value and then map this value back to a set of base type values, thus covering a larger scope.

Let $x^{(i)}$ be a value of type $T^{(i)}$, where $i = 0, 1, \ldots, k$. Since each value corresponds to a node in the TAHs, moving from one node $x^{(i)}$ up one level to another node $x^{(i+1)}$ in a TAH

30

can be accomplished by the function $generalize(x^{(i)})$ and moving from a node $x^{(i)}$ down to the bottom $x^{(0)}$ can be accomplished by $specialize(x^{(i)})$. The result of $generalize(x^{(i)})$ is always a single value and of a type more abstract than the type of $x^{(i)}$ (*i.e.*, some $x^{(i+1)}$), and values in $specialize(x^{(i)})$ will be the leaf nodes (*i.e.*, the value of the base type $T^{(0)}$) as defined recursively,

$$specialize(x^{(i)}) = \begin{cases} \{x^{(0)}\} & \text{if } i = 0 \ (i.e., \text{ the leaf node}) \\ \bigcup_{\forall x^{(i-1)}} specialize(x^{(i-1)}) & \text{otherwise} \end{cases}$$

where $x^{(i-1)} \in \{x^{(i-1)} | x^{(i)} = generalize(x^{(i-1)})\}$.

Then a single step relaxation for a value $x^{(i)}$ is defined as

$$relax(x^{(i)}) = specialize(generalize(x^{(i)}))$$

For a value $x^{(i)}$ of type $T^{(i)}$, it can be relaxed $k - i$ times at most. A j-step relaxation (where $j = 1, 2, \ldots, k$) can be obtained by applying $generalize()$ j times and then applying $specialize()$ once to the result:

$$relax^{(j)}(x^{(i)}) = specialize(generalize^j(x^{(i)})), \ where \ i \leq k - j$$

Relaxation has the following property:

$$relax^{(j)}(x^{(i)}) \subseteq relax^{(j+1)}(x^{(i)}), for \ i = 0, \ldots, k - 2 \ and \ j = 1, \ldots, k - i - 1.$$

The above definition is also applicable to cases where the base types are tuple types (*i.e.*, MTAHs).

3. Automatic Knowledge Acquisition

The automatic generation of knowledge base (TAHs) from databases is essential for CoBase to be scalable to large systems. We have developed algorithms to automatically generate TAHs based on database instances. A brief discussion about the algorithms and their complexity are followed.

3.1. Numerical TAHs

COBWEB (Fisher, 1987), a conceptual clustering system, uses *category utility* (CU) (Gluck and Corter, 1985) as a quality measure to classify the objects described by a set of attributes into a classification tree. Formally, for a partition from a class C to N mutually exclusive classes $C_1, \ldots, C_N$, the category utility (CU) is defined as the increase in the *goodness*[2] of these classes after partition. That is,

$$CU(C_1, \ldots, C_N) = \frac{\sum_{k=1}^{N} P(C_k)G(C_k) - G(C)}{N} \tag{1}$$

where $P(C_k)$ is the occurrence probability of C_k in C, and $G(C_k)$ and $G(C)$ are the goodness functions for C_k and C, respectively. That is,

$$G(C_k) = \sum_{a \in A} \sum_{x_i^a \in X_k^a} P(x_i^a)^2 \qquad (2)$$

$$G(C) = \sum_{a \in A} \sum_{x_i^a \in X^a} P(x_i^a)^2 \qquad (3)$$

where A is the set of all the attributes, and X_k^a and X^a are the distinct values of attribute a in C_k and C respectively.

COBWEB cannot be used for abstracting numerical data; it only deals with categorical data. Moreover, its classification tree serves as *the database* for instances and requires a large storage space. Furthermore, matching of objects with existing classes is time consuming. For providing approximate answers, we want to build a classification tree that minimizes the difference between the desired answer and the derived answer. Specifically, we use relaxation error as a measure for clustering. The *relaxation error* (RE) is defined as the average difference between the requested values and the returned values. $RE_1(C)$ can also be interpreted from the standpoint of query relaxation. Let us define the *relaxation error of* x_i, $RE_1(x_i)$, as the average difference from x_i to x_j, $j=1,...,n$, that is,

$$RE_1(x_i) = \sum_{j=1}^{n} P(x_j)|x_i - x_j| \qquad (4)$$

where $P(x_j)$ is the occurrance probability of x_j in C. $RE_1(x_i)$ can be used to measure the quality of an approximate answer where x_i in a query is relaxed to x_j, $j=1,...,n$. Summing $RE_1(x_i)$ over all values x_i in C, we have

$$RE_1(C) = \sum_{i=1}^{n} P(x_i)RE_1(x_i). \qquad (5)$$

Thus, $RE_1(C)$ is the expected error of relaxing any value in C.

If $RE_1(C)$ is large, query relaxation based on C may produce very poor approximate answers. To overcome this problem, we can partition C into sub-clusters to reduce relaxation error. Given a partition $P = \{C_1, C_2, ..., C_N\}$ of C, the *relaxation error of the partition P* is defined as

$$RE_1(P) = \sum_{k=1}^{N} P(C_k)RE_1(C_k) \qquad (6)$$

where $P(C_k)$ equals the number of tuples in C_k devided by the number of tuples in C. In general, $RE_1(P) < RE_1(C)$.

Relaxation error is the expected pair-wise difference between values in a cluster. To extend the notion of relaxation error from a single attribute to multiple attributes, we shall consider

distance between tuples instead of difference between values. Given two m-attribute tuples $t_i = \{x_1, ..., x_m\}$ and $t_j = \{y_1, ..., y_m\}$, their distance is defined as

$$D(t_i, t_j) = \sum_{k=1}^{m} W_k \frac{|x_k - y_k|}{\Delta_k} \tag{7}$$

where W_k and Δ_k are the weight and the normalization constant for the k-th attribute, respectively. W_k can be assigned to reflect relative importance among attributes. Δ_k is necessary for summing up differences from different attributes because these attributes have different distributions.

Given a cluster of n m-attribute tuples $C = \{t_1, ..., t_n\}$, the relaxation error for C is defined as the average pair-wise distance among tuples in C, that is,

$$RE(C) = \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{n} P(t_i) P(t_j) D(t_i, t_j) \tag{8}$$

where $P(t_i)$ and $P(t_j)$ are the probabilities of tuples t_i and t_j, respectively. The division by m in (8) normalizes RE(C) per attribute and allows us to compare relaxation errors computed from different numbers of attributes. The category utility (CU) for multiple attributes can be obtained by simply substituting $RE_1(C)$ in Eq (9) by its multi-attribute counterpart $RE(C)$ in Eq (8).

Using relaxation error, the category utility can be defined as the relaxation error reduction per sub-cluster, that is,

$$\begin{aligned} CU &= \frac{\sum_{k=1}^{N} P(C_k)[1 - RE_1(C_k)] - [1 - RE_1(C)]}{N} \\ &= \frac{RE_1(C) - \sum_{k=1}^{N} P(C_k) RE_1(C_k)}{N}. \end{aligned} \tag{9}$$

DIstribution Sensitive Clustering (DISC) (Chu and Chiang, 1994) partitions sets of numerical values into clusters that minimize the relaxation error. We shall now present a class of DISC algorithms for clustering numerical values. We shall present the algorithm for a single attribute, and then extend it for multiple attributes.

The Clustering Algorithm for a Single Attribute

Given a cluster with n distinct values, the number of partitions is exponential with respect to n, so the best partition according to (9) takes exponential time to find. To reduce computation complexity, we shall only consider binary partitions (i.e., $N = 2$ in (9)). Later we shall show a simple hill climbing strategy can be used for obtaining N-ary partitions from binary partitions.

Our method is top down: we start from one cluster consisting of all the values of an attribute, and then we find "cuts"[3] to recursively partition the cluster into smaller clusters. The partition result is a concept hierarchy called Type Abstraction Hierarchy (TAH). The clustering algorithm is called the DISC (DIstribution Sensitive Clustering) Method and is given in Table 1.

Algorithm DISC(C)
    if the number of distinct values $\in C < T$, return /* T is a threshold */
    let *cut* = the best cut returned by **BinaryCut(C)**
    partition values in C based on *cut*
    let the resultant sub-clusters be C_1 and C_2
        call DISC(C_1) and DISC(C_2)

Algorithm BinaryCut(C)
/* input cluster $C = \{x_1, ..., x_n\}$ */
    for $h = 1$ to $n - 1$ /* evaluate each cut */
        Let P be the partition with clusters $C_1 = \{x_1, ..., x_h\}$ and $C_2 = \{x_{h+1}, ..., x_n\}$
        compute $RE_1(P)$
        if $RE_1(P) < MinRE$ then
            $MinRE = RE_1(P)$, $cut = h$ /* the best cut */
    Return *cut* as the best cut

Table 1. The algorithms DISC and BinaryCut

In (Chu and Chiang, 1994), an implementation of the algorithm BinaryCut is presented whose time complexity is $O(n)$. Since DISC needs to execute BinaryCut $n - 1$ times at most to generate a TAH, the worst case time complexity of DISC is $O(n^2)$. (The average case time complexity of DISC is $O(n \log n)$.)

N-ary Partitioning

N-ary partitions can be obtained from binary partitions by a hill climbing method. Starting from a binary partition, the sub-cluster with greater relaxation error is selected for further cutting. We shall use CU as a measure to determine if the newly formed partition is better than the previous one. If the CU of the binary partition is greater than that of the tri-nary partition, then the tri-nary partition is dropped and the cutting is terminated. Otherwise, the tri-nary partition is selected and the cutting process continues until it reaches the point where a cut decreases CU. The procedure is outlined in Table 2.

The Clustering Algorithm for Multiple Attributes

Query relaxation for multiple attributes using multiple single-attribute TAHs relaxes each attribute independently disregarding the relationships that might exist among attributes. This may not be adequate for the applications where attributes are dependent.[4] In addition, using multiple single attribute TAHs is inefficient since it may need many iterations of query modification and database access before approximate answers are found. Furthermore, relaxation control for multiple TAHs is more complex since there is a large number of possible orders for relaxing attributes. In general, we can only rely on simple heuristics such as *best first* or *minimal coverage first* to guide the relaxation (see Section 5.2.1). These heuristics cannot guarantee best approximate answers since they are rules of thumb and not necessarily accurate.

```
Algorithm N-ary Partition(C)
let C_1 and C_2 be the two sub-clusters of C
compute CU for the partition C_1, C_2
for N = 2 to n - 1
        let C_i be the sub-cluster of C with maximum relaxation error
        call BinaryCut to find the best sub-clusters C_{i1} and C_{i2} of C_i
        compute and store CU for the partition C_1, ..., C_{i-1}, C_{i1}, C_{i2}, C_{i+1}, ..., C_N
        if current CU is less than the previous CU
                stop
        else
                replace C_i by C_{i1} and C_{i2}
/* the result is an N-ary partition of C */
```

Table 2. The N-ary partition algorithm

```
Algorithm M-DISC(C)
        if the number of objects in C < T, return /* T is a threshold */
        for each attribute a = 1 to m
                for each possible binary cut h
                        compute CU for h
                        if CU > MaxCU then /* remember the best cut */
                            MaxCU = CU, BestAttribute = a, cut = h
                partition C based on cut of the attribute BestAttribute
                let the resultant sub-clusters be C_1 and C_2
                        call M-DISC(C_1) and M-DISC(C_2)
```

Table 3. Multi-attribute DISC (M-DISC) algorithm

Most of the above mentioned difficulties can be overcome by using Multi-attribute TAH (MTAH) for the relaxation of multiple attributes. Since MTAHs are generated from *semantically dependent* attributes, these attributes are relaxed together in a single relaxation step, thus greatly reducing the number of query modification and database access. Approximate answers derived by using MTAH have better quality than those derived by using multiple single-attribute TAHs. MTAHs are context and user sensitive because a user may generate several MTAHs with different attribute sets from a table. Should a user needs to create an MTAH containing semantically-dependent attributes from different tables, these tables can be joined into a single view for MTAH generation.

To cluster objects of multiple attributes, DISC can be extended to M-DISC (shown in Table 3). The generated multi-dimensional TAHs are called MTAHs. The algorithm DISC is a special case of M-DISC, and TAH is a special case of MTAH. Let us now consider the time complexity of M-DISC. Let m be the number of attributes and n be the number of distinct attribute values. The computation of relaxation error for a single attribute takes $O(n \log n)$ to complete (Chu and Chiang, 1994). Since the computation of CU involves computation of relaxation error for m attributes, its complexity is $O(mn \log n)$. The nested loop in M-DISC is executed mn times, so the time complexity of M-DISC is $O(m^2 n^2 \log n)$. To

generate an MTAH, it takes no more than n calls of M-DISC, therefore, the worst case time complexity of generating an MTAH is $O(m^2 n^3 \log n)$. The average case time complexity is $O(m^2 n^2 (\log n)^2)$ since M-DISC needs only to be called $\log n$ times on the average.

3.2. Non-numerical TAHs

For generating TAHs with non-numerical attributes, the values are clustered based on inter-attribute relationships. A Pattern-Based Knowledge Induction (PBKI) algorithm (Merzbacher and Chu, 1993) is developed to compute the pair-wise correlations among attribute values. Correlation is then used as a nearness measure in clustering the attribute values. A more detailed discussion about the algorithm is presented in (Merzbacher and Chu, 1993).

3.3. Maintenance of TAHs

Since the quality of TAH affects the quality of derived approximate answers, TAHs should be kept up to date. One simple way for maintaining TAHs is to regenerate them whenever an update occurs. This approach is not desirable because it causes overhead for the database system. Although each update changes the distribution of data (thus changing the quality of the corresponding TAHs), this may not be significant enough to warrant a TAH regeneration. TAH regeneration is only necessary when the cumulative effect of updates has greatly degraded the TAHs. The quality of a TAH can be monitored by comparing the derived approximate answers to the expected relaxation error (for example, see Figure 8) which is computed at TAH generation time and recorded at each node of the TAH. When the derived approximate answers significantly deviate from the expected quality, then the quality of the TAH is deemed to be inadequate and a regeneration is necessary. The following incremental TAH regeneration procedure can be used. First, identify the node within the TAH that has the worst query relaxations. Apply partial TAH regeneration for all the database instances covered by the node. After several such partial regenerations, we then initiate a complete TAH regeneration.

The generated TAHs are stored in Unix files and a TAH Manager (described in Section 5.3) is responsible to parse the files, create internal representation of TAHs, and provide operations such as generalization and specialization to traverse TAHs. The TAH Manager also provides a directory that describes the characteristics of TAHs (e.g., attributes, names, user type, context, TAH size, location) for the users/systems to select the appropriate TAH to be used for relaxation.

Our experience in using DISC/MDISC and PBKI for ARPI transportation data-bases (94 relations, the biggest one of which has 12 attributes and 195,598 tuples) shows that the clustering techniques for both numerical and non-numerical attributes can be generated from a few seconds to a few minutes depending on the table size on a Sun SPARC 20 Workstation.

4. Cooperative SQL (CoSQL)

4.1. CoSQL Cooperative Operators

The cooperative operations consist of the following four types: context free, context sensitive, control, and interactive. The context free and context sensitive cooperative operators can be used in conjunction with attribute values specified in the WHERE clause. The relaxation control operators can only be used on attributes specified in the WHERE clause, and the control operators have to be specified in the WITH clause after the WHERE clause. The interactive operators can be used alone as command inputs.

Context free operations.

- *Approximate operator*, $\wedge v$, relaxes the specified value v within the approximate range that is predefined by the user. For example, $\wedge$9am transforms into the interval (8am, 10am).

- *Between* (v_1, v_2) specifies the interval for an attribute. For example, time between (7am, $\wedge$9am) transforms into (7am, 10am). The transformed interval is pre-specified either by the user or the system.

- *Within* $(x_1, y_1, ..., x_n, y_n)$ specifies a region on a map by a closed polygon $(x_1, Y_1, ..., x_n, y_n)$.

Context sensitive.

- *Near-to X* is used for specification of spatial nearness of object X. The "near-to" measure is context and user sensitive. "Nearness" can be determined by the TAHs or specified by the user. For example, near-to 'BIZERTE' requests the list of cities in the same cluster as BIZERTE in a TAH, or cities located within a certain Euclidean distance (depending on the context) from the city 'BIZERTE'.

- *Similar-to X based-on* $((a_1 \ w_1)(a_2 \ w_2)...(a_n \ w_n))$ is used to specify a set of objects semantically similar to the target object X based on a set of attributes $(a_1, a_2, ..., a_n)$ specified by the user. Weights $(w_1, w_2, ..., w_n)$ may be assigned to each of the attributes to reflect the relative importance in considering the similarity measure. The set of similar objects can be ranked by the similarity. The similarity measures that computed from the nearness (e.g. weighted mean square error) of the pre-specified attributes to that of the target object. The set size is bound by a pre-specified minimum answer set size.

Control Operators.

- *Relaxation-order* $(a_1, a_2, ..., a_n)$ specifies the order of the relaxation among the attributes $(a_1, a_2, ..., a_n)$ (*i.e.*, a_i precedes a_{i+1}). For example, relaxation-order (runway_length, runway_width) indicates that if no exact answer is found, then runway_length should be relaxed first. If still no answer is found, then relax the runway_width. If no relaxation-order control is specified, the system relaxes according to its default relaxation strategy.

- *Not-relaxable* $(a_1, a_2, ..., a_n)$ specifies the attributes $(a_1, a_2, ..., a_n)$ that should not be relaxed. For example, `not-relaxable location_name` indicates that the condition clause containing location_name must not be relaxed.

- *Preference-list* $(v_1, v_2, ..., v_n)$ specifies the preferred values $(v_1, v_2, ..., v_n)$ of a given attribute, where v_i is preferred over v_{i+1}. As a result, the given attribute is relaxed according to the order of preference that the user specifies in the preference list. For example, the attribute "food style," a user may prefer Italian food to Mexican food. If there are no such restaurants within the specified area, the query can be relaxed to include the foods similar to Italian food first, and then similar to Mexican food.

- *Unacceptable-list* $(v_1, v_2, ..., v_n)$ allows users to inform the system not to provide certain answers. This control can be accomplished by trimming parts of the TAH from searching. For example, "avoid airlines X and Y" tells the system that airlines X and Y should not be considered during relaxation. It not only provides more satisfactory answers to users, but also reduces search time.

- *Use-TAH (TAH-name)* allows users to specify the TAHs of their choices. For example, a vacation traveler may want to find an airline based on its fare while a business traveler is more concerned with his schedule. To satisfy the different needs of the users, several TAHs of airlines can be generated, emphasizing different attributes (*e.g.*, price and nonstop flight).

- *Relaxation-level* (v) specifies the maximum allowable range of the relaxation on an attribute, *i.e.*, $[0, v]$.

- *At least* (s) specifies the minimum number of answers required by the user. CoBase relaxes query conditions until enough number of approximate answers (*i.e.*, $\geq s$) are obtained.

User/System interaction operators.

- *Nearer, Further* provide users with the ability to control the "near-to" relaxation scope interactively. Nearer reduces the distance by a pre-specified percentage while further increases the distance by a pre-specified percentage.

4.2. Editing Relaxation Control Parameters

Users can browse and edit relaxation control parameters to better suit their applications (for example, Figure 2). The parameters include the relaxation range for the "approximately-equal" operator, the default distance for the "near-to" operator, the number of returned tuples for the "similar-to" operator, *etc.*

4.3. Examples

In this section, we present a few selected examples that illustrate the capabilities of the cooperative operators. The corresponding TAHs used for query modification are shown in Figure 1 and the default relaxable ranges are shown in Figure 2.

Query 1. List all the airports with the runway length greater than 7500 feet and runway width greater than 100 feet. If there is no answer, relax the runway length condition first. The following is the corresponding CoSQL query:

```
SELECT aport_name, runway_length_ft, runway_width_ft
FROM aports
WHERE runway_length_ft > 7500 AND runway_width_ft > 100
WITH  RELAXATION-ORDER (runway_length_ft, runway_width_ft)
```

Based on the runway-length TAH and the relaxation order, the query is relaxed to:

```
SELECT aport_name, runway_length_ft, runway_width_ft
FROM aports
WHERE runway_length_ft >= 4000 AND runway_width_ft > 100
```

If this query yields no answer, then we proceed to relax the range of runway width.

Query 2. Find all the airports with their geographical coordinates near Bizerte in the country Tunisia. If there is no answer, the restriction on the country should not be relaxed. The corresponding CoSQL query is as follows:

```
SELECT aport_name, latitude, longitude
FROM aports, GEOLOC
WHERE aport_name NEAR-TO 'Bizerte'
      AND country_state_name = 'Tunisia'
      AND GEOLOC.geo_code = aports.geo_code
WITH  NOT-RELAXABLE country_state_name
```

Based on the TAH on airport location in Tunisia (Figure 1), the relaxed query is:

```
SELECT aport_name, latitude, longitude
FROM aports, GEOLOC
WHERE aport_name IN ('Bizerte', 'Djedeida',
                     'Saminjah', 'Tunis')
      AND GEOLOC.geo_code = aports.geo_code
```

If the TAH for the airport location in Tunisia is not available, the system defaults to a pre-defined range (Figure 2) which relaxes the search range by 100 miles from Bizerte.

Query 3. Find all airports in Tunisia similar to the Bizerte airport. Use the attributes runway_length_ft, runway_width_ft as criteria for similarity. Place more similarity emphasis on runway length than runway width; their corresponding weight assignments are 2 and 1, respectively. The following is the CoSQL version of the query:

Approximate Operator Relaxation Range

Relation Name	Attribute Name	∧Range
aports	runway_width_ft	10
aports	parking_sq_ft	100000

Near-to Operator Relaxation Range

Relation Name	Attribute Name	Near-to Range	Near/Further
aports	aport_name	100 miles	50%

Figure 2. Relaxation Range for the Approximate and Near-to Operators.

```
SELECT aport_name
FROM aports, GEOLOC
WHERE aport_name SIMILAR-TO 'Bizerte'
                 BASED-ON ((runway_length_ft 2.0)
                           (runway_width_ft 1.0))
      AND country_state_name = 'TUNISIA'
      AND GEOLOC.geo_code = aports.geo_code
```

To select the set of the airport names that have the runway length and runway width similar to the Bizerte airport, we shall first find the values of the attributes: runway_length_ft and runway_width_ft for the Bizerte airport,therefore transform the query to:

```
SELECT runway_length_ft, runway_width_ft
FROM aports, GEOLOC
WHERE country_state_name_ = 'TUNISIA'
      AND GEOLOC.geo_code = aports.geo_code
      AND aport_name='Bizerte'
```

The system relaxes the values of the attributes: runway_length_ft and runway_width_ft using an MTAH or TAHs until the size of the similarity set is satisfied by the pres-specified answer set size. The system then computes the similarity of these airports to "Bizerte" using the pre-specified *nearness* formula (*e.g.*, weighted mean squared error). The order in the similarity set is ranked according to the nearness meausre.

5. A Scalable and Extensible Architecture

Figure 3 shows an overview of the CoBase System. Type abstraction hierarchies and relaxation ranges for the explicit operators are stored in a knowledge base (KB). There is a TAH directory storing the characteristics of all the TAHs in the system. When CoBase asks queries, it asks the underlying database systems (DBMS). When an approximate answer
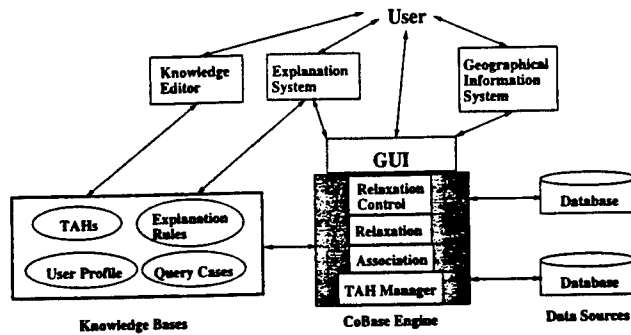
Figure 3. CoBase Functional Architecture.

is returned, the user can ask for an explanation of how the answer was derived or an annotated relaxation path. A context-based semantic nearness will be provided to rank the approximate answers (in order of nearness) against the specified query. A GUI displays the query, results, TAHs, and explanations of the relaxation processes. Based on user type and query context, associative information is derived from past query cases. A user can construct TAHs from one or more attributes and modify the existing TAH in the KB.

5.1. Mediator Architecture

We use the concept of *mediation* (Wiederhold, 1992) to decompose our cooperative answering system into reusable and interacting components. A *Mediator* is a software module that takes some input set of information, intelligently analyzes the information from a specific viewpoint, and produces a set of conclusions based on its analysis. Often, a Mediator needs additional information (knowledge and/or data) to perform its analysis. This information can be required as inputs to the Mediator's analysis, the Mediator can seek the assistance of other Mediators in fulfilling its information needs. This latter mode (Mediators assisting Mediators) is called *dynamic matching*. Specifically, a Mediator with an information need will report this need to the environment and dynamically match with the Mediators that can fulfill the need.

Our use of Mediators is to decompose the cooperative query answering capabilities, where:

- Each mediator is composed of (1) its mediation process; (2) its mediation capabilities specifying what the mediator produces given a specific input set; and (3) its mediation requirements specifying what information the mediator needs access during processing (information that is not the postcondition input).
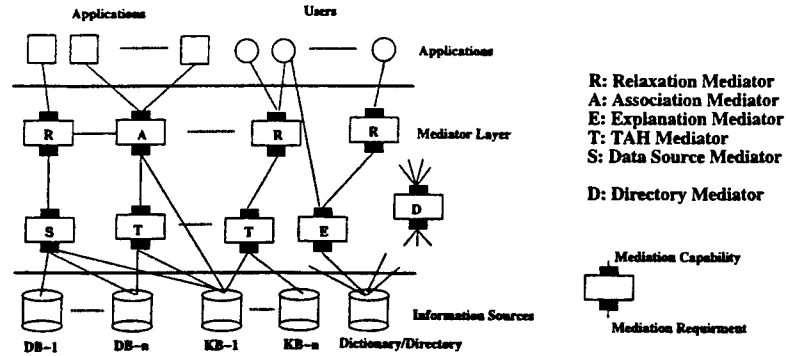
41

Figure 4. A Scalable and Extensible Cooperative Information System

- One mediator's requirements can match (or link to) another mediator's capabilities – this represents the linking mediator's matching with the linked to mediator's information process.

- One or more *Directory Mediators* are needed to act as the information repository of the mediator set. Specifically, the directory mediator catalogs the mediators available in the system by their capabilities and requirements. It can then be consulted to find a mediator which meets a specific information need.

Figure 4 displays the various cooperative mediators: Relaxation, TAH, Association, Explanation, Data Source, and Directory. These mediators are connected selectively to meet applications' needs. An application that requires relaxation and explanation capabilities, for example, will entail a linking of Relaxation and Explanation mediators. Our mediator architecture allows incremental growth with application. When the demand for certain mediators increases, additional copies of the mediators can be added to reduce the loading, thus the system is scalable. For example, there are multiple copies of relaxation mediator and TAH mediator in Figure 4. Further, different types of mediators can be interconnect together and communicate with each other via a common communication protocol (*e.g.*, KQML) to perform a joint task. Thus, the architecture is extensible.

5.2. Relaxation Mediator

Figure 5 illustrates the functional components of the relaxation mediator, its capability, and input requirements. Query relaxation is the process of understanding the semantic context, intent of a user query and modifying the query constraints with the guidance of the customized knowledge structure (TAH) into "near" values that provide "best-fit" answers. The flow of the relaxation process is depicted in Figure 6. When a CoSQL query is presented to the Relaxation Mediator, the system first go through a pre-processing phase. During the pre-processing, the system first relaxes any context free and/or context
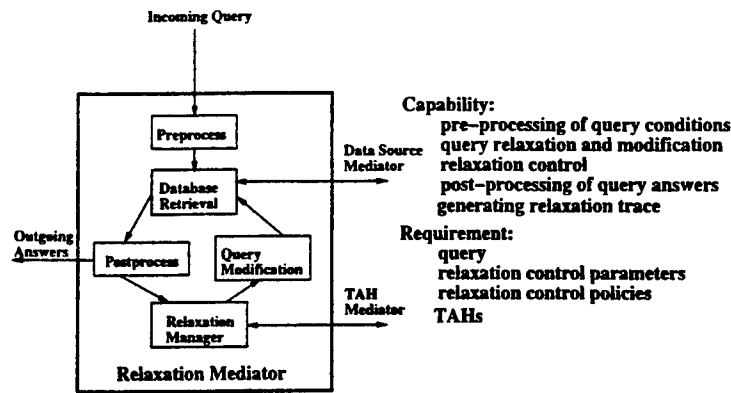
Figure 5. The Relaxation Mediator
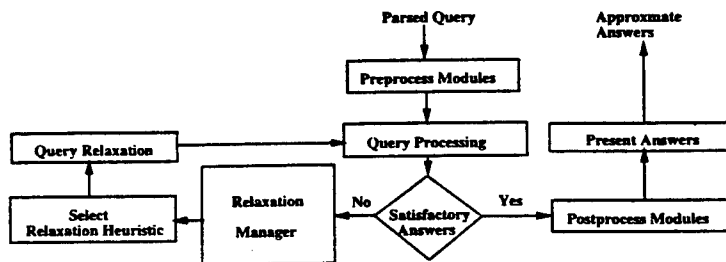
Figure 6. Flow chart for Processing CoBase queries.

sensitive cooperative operators in the query. All relaxation control operations specified in the query will be processed. The information will be stored in the relaxation manager and ready to be used if the query requires relaxation. The modified SQL query is then presented to the underlying database system for execution. If no answers are returned, then the cooperative query system, under the direction of the Relaxation Manager, relaxes the queries by query modification. This is accomplished by traversing along the TAH node for performing generalization and specialization and rewriting the query to include a larger search scope. The relaxed query is then executed, and if there is no answer, we repeat the relaxation process until we obtain one or more approximate answers. If the system fails to produce an answer due to over-trimmed TAHs, the relaxation manager will deactivate certain relaxation rules to restore part of a trimmed TAH to broaden the search scope until answers are found. Finally, the answers are post-processed (*e.g.*, ranking, filtering, *etc*).

5.2.1. Relaxation Control

Relaxation without control may generate more approximations than the user can handle. The policy for relaxation control depends on many factors, including user profile, query context, and relaxation control operators as defined in Section 4.1. The Relaxation Manager combines those factors via certain policies (minimizing search time or nearness, for example) to restrict the search for approximate answers. We allow the input query to be annotated with control operators to help guide the mediator in query relaxation operations.

If control operators are used, the Relaxation Manager selects the condition to relax in accordance with the requirements specified by the operators. For example, a "relaxation-order" operator will dictate "relax location first, then runway length." Without such user-specified requirements, the Relaxation Manager uses a default relaxation strategy by selecting the relaxation order based on the *minimum coverage rule*. Coverage is defined as the ratio of the cardinality of the set of instances covered by the entire TAH. Thus, *coverage* of a TAH node is the percentage of all tuples in the TAH which are covered by the current TAH node. The minimum coverage rule always relaxes the condition which causes the minimum increase in the scope of the query, which is measured by the coverage of its TAH node. This default relaxation strategy attempts to add the smallest number of tuples possible at each step, based on the rationale that the smallest increase in scope is likely to generate the close approximate answers. The strategy for choosing which condition to be relaxed first is only one of many possible relaxation strategies; the Relaxation Manager can support other different relaxation strategies as well.

Let us consider the following example of using control operators to improve the relaxation process. Suppose a pilot is searching for an airport with an 8,000 feet runway in Bizerte but there is no airport in Bizerte that meets the specifications. There are many ways to relax the query in terms of location and runway length. If the pilot specifies the relaxation-order to relax the location attribute first, then the query modification generalizes the location 'Bizerte' to 'NW_Tunisia' (as shown in Figure 1) and specializes the locations 'Bizerte', 'Djedeida', 'Tunis', and 'Saminjah', thus broadening the search scope of the original query. If, in addition, we know that the user is interested only in the airports in NW and SW Tunisia and does not wish to shorten the required runway length, the system can eliminate the search in East Tunisia and also avoid airports with short and medium runways, as shown in Figure 7. As a result, we can limit the query relaxation to a narrower scope by trimming the type abstraction hierarchies, thus improving both the system performance and the answer relevance.

5.2.2. Spatial Relaxation and Approximation

In geographical queries, spatial operators such as 'located,' 'within,' 'contain,' 'intersect,' 'union,' and 'difference' are used. When there are no exact answers for a geographical query, both its spatial and non-spatial conditions can be relaxed to obtain the approximate answers. CoBase operators also can be used for describing approximate spatial relationships. For instance, "an aircraft-carrier is *near* seaport Sfax." *Approximate spatial operators*, such as 'near-to' and 'between' are developed for the ap-
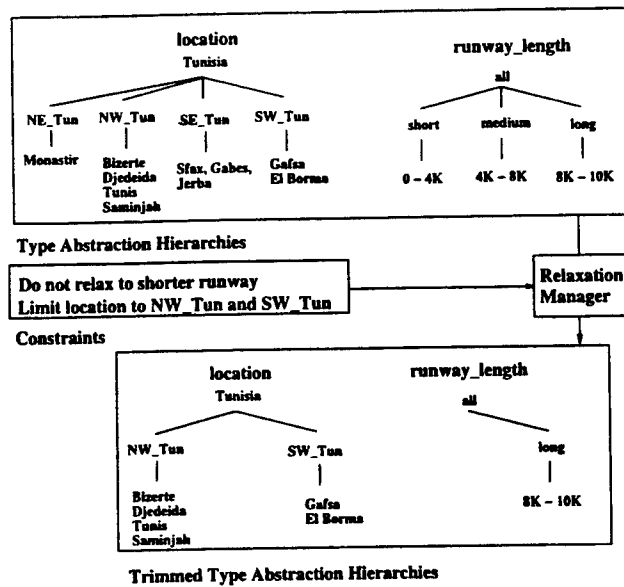
Figure 7. TAH trimming based on relaxation control operator.

proximate spatial relationships. Spatial approximation depends on *contexts* and *domains* (Mark and Frank, 1989, Subramanian and Adam, 1993). For example, a **hospital** *near to* LAX is different from an **airport** *near to* LAX. Likewise, the *nearness* of a hospital in a **metropolitan** area is different from the one in a **rural** area. Thus spatial conditions should be relaxed differently in different circumstances. A common approach to this problem is the use of pre-specified ranges. This approach requires experts to provide such information for all possible situations, which is difficult to scale up to larger applications or to extend to different domains. Since TAHs are user and context sensitive, they can be used to provide context-sensitive approximation. More specifically, we can generate TAHs based on multi-dimensional spatial attributes (MTAHs).

Further, MTAH (based on latitude and longitude) is generated based on the distribution of the object locations. The distance between nearby objects is context-sensitive: the denser the location distribution, the smaller the distance among the objects. In Figure 8, for example, the default neighborhood distance in Area 3 is smaller than the one in Area 1. Thus, when a set of airports are clustered based on their locations, the ones in the same cluster of the MTAH are much closer to each other than to those outside the cluster. Thus, they can be considered 'near-to' each other. We can apply the same approach to other approximate spatial operators, such as 'between' (*i.e.*, a cluster 'near-to' the center of two objects). MTAHs also can be used to provide context-sensitive query relaxation. For example, consider the query: "Find an airfield at the city Sousse." Since there is no airfield located exactly at Sousse, this query therefore can be relaxed to obtain approximate answers. First, we locate the city Sousse with latitude 35.83 and longitude 10.63. Using
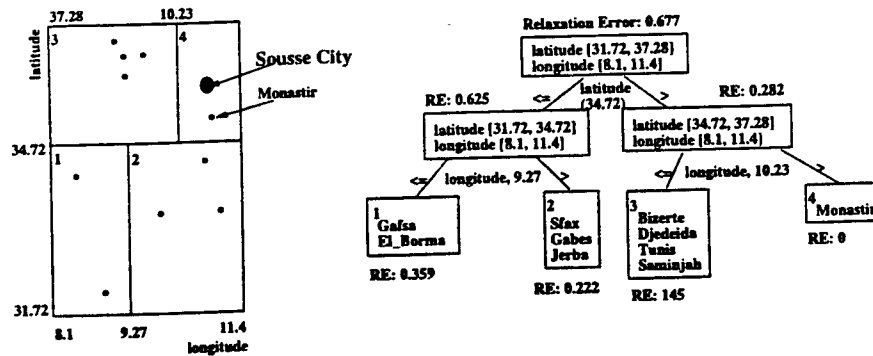
Figure 8. An MTAH for the airports in Tunisia and its corresponding two-dimensional space.

the MTAH in Figure 8, we find Sousse is covered by Area 4. Thus, the airport Monastir is returned. Unfortunately, it is not an airfield. So the query is further relaxed to the neighboring cluster–the four airports in Area 3 are returned: Bizerte, Djedeida, Tunis, and Saminjah. Since only Djedeida and Saminjah are airfields, these two will be returned as the approximate answers.

MTAHs are automatically generated from databases by using our clustering method that minimizes relaxation error (Chu and Chiang, 1994). They can be constructed for different contexts and user type. For example, it is critical to distinguish a friendly airport from an enemy airport. The use of a MTAH for friendly airports restricts the relaxation only within the set of friendly airports, even though some enemy airports are geographically nearby. This restriction significantly improves the accuracy and flexibility of spatial query answering. The integration of spatial and cooperative operators provides more expressiveness and context-sensitive answers. For example, the user is able to pose such queries as, "find the airports similar-to LAX and near-to City X." When there are no answers available, both 'near-to' and 'similar-to' can be relaxed based on the user's preference (*i.e.*, a set of attributes). To relax 'near-to', airports from neighboring clusters in the MTAH are returned. To relax 'similar-to' the multiple-attribute criteria are relaxed by their respective TAHs.

Cooperativeness in geographic databases was studied in (Hemerly et al., 1993). A rule-based approach is used in their system for approximate spatial operators as well as query relaxation. For example, they define that: "P is *near* to Q iff the distance from P to Q is less than $n*length\_unit$, where *length_unit* is a context dependent scalar parameter, and n is a scalar parameter that can be either unique for the application and thus defined in domain model, or specific for each class of users and therefore defined in the user models." This approach requires n and *length_unit* be set by domain experts. Thus, it is difficult to scale up. Our system uses MTAHs as a representation of the domain knowledge. The MTAHs can be generated automatically from databases based on contexts and provide a structured and context-sensitive way to relax queries. As a result, it is scalable to large applications. Further, the relaxation error at each node is computed during the construction of TAHs and
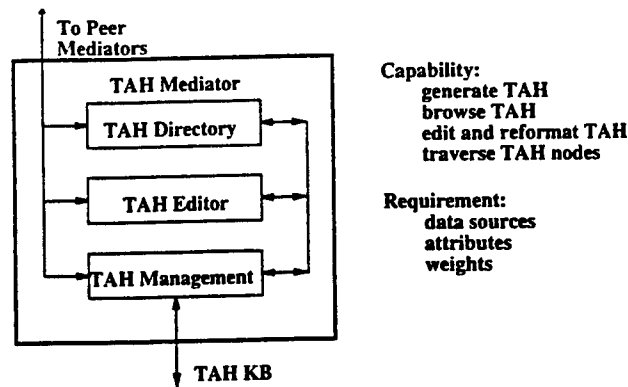
Figure 9. The TAH Mediator

MTAHs. It can be used to evaluate the quality of relaxations and to rank the nearness of the approximate answers to the exact answer.

5.3. TAH Mediator

The CoBase TAH Mediator (Figure 9) provides three conceptually separate, yet interlinked, functions to peer mediators. These are the TAH Directory, the TAH Management, and the TAH Editing facilities.

Usually, a system contains a large number of TAHs. In order to allow other mediators to determine which TAHs exist within the system and the characteristics of those TAHs, the TAH Mediator contains an intelligent directory. This directory serves to link the characteristics, context, and user type with the TAHs themselves. This directory is searchable by characteristics, user type, context, name, or any combination thereof, enabling peer mediators to locate TAHs even if they have only a partial picture of the TAH they desire. Further, the TAH directory is capable of approximate matching, so if the exact TAH a client desires is unavailable in the system, the TAH Directory lists those TAHs which may be useful.

The TAH Management facility provides client mediators with TAH traversal functions (e.g. specialization and generalization), data extraction functions (for reading the information out of TAH nodes), and formatting functions (to eliminate the need for peer mediators to understand the varied internal formats of different types of TAHs). These capabilities present a common interface, so that peer mediators can traverse and extract data from a TAH without knowing about the particular type or structure of the TAHs they are using.

The TAH Mediator supports a TAH editor which allows users to edit TAHs to suit their specific needs. The editor provides the following editing operations.

- **Delete:** to improve search efficiency, irrelevant parts of a hierarchy for a given application should be deleted.-

- **Move:** if a TAH is improperly formed for its application, then a domain expert can adjust it by moving sub-hierarchies and attaching them to the appropriate places.

47
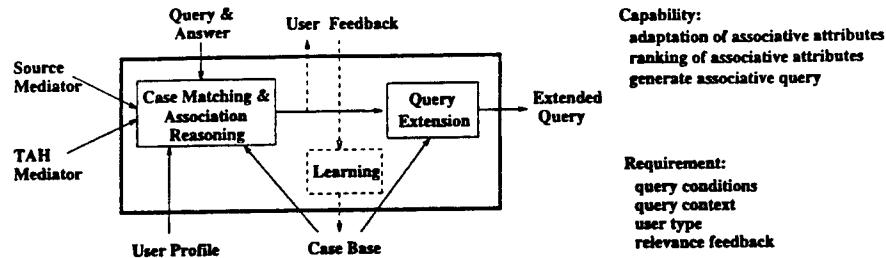
Figure 10. Association Mediator

- **Add:** if a TAH does not provide enough details for the particular application, then additional information can be added to the hierarchy.

The TAH editor handles recalculation of all information contained within TAH nodes (e.g. the coverage and relaxation error) during the editing process and supports exportation and importation of entire TAHs if a peer mediator wishes to modify a TAH itself.

5.4. Association Mediator

Often it is desirable to provide additional information relevant to, though not explicitly stated in, a user's query. For example, in finding the location of an airport satisfying the runway length and width specifications, an association mediator (Figure 10) can provide additional information about the runway quality and weather condition so that this additional information may help the pilot select a suitable airport to land his aircraft. Thus, association needs to identify the dependencies and relationships between data distributed in multiple classes or even multiple databases. Since objects have different relationships in different problem domains, it is necessary to focus on the localized contexts in which the objects participate. Domain knowledge is then used to interpret the relationships between these objects. As an object can have multiple super-types, there exist different views of type abstraction for different problem contexts. Therefore, TAHs express relationships at the context level.

Association in CoBase is executed as a multi-step post-process. After the query is executed, the answer set is gathered with the query conditions, user profile, and application constraints. This combined information is matched against query cases from the case base to identify relevant associative information (Fouque et al., 1994). The query cases can take the form of a CoBase query which can include any CoBase construct, such as conceptual conditions (*e.g.*, runway_length_ft = short) or explicitly cooperative operations (city *near-to* 'Bizerte').

48

Query Answer		Associative Information	
name	runway_length	runway_condition	weather
Jerba	9500	Damaged	Sunny
Monastir	6500	Good	Foggy
Tunis	8500	Good	Good

Figure 11. Query Answer and Associative Information for the Selected Airports.

For example, consider the query

```
SELECT name, runway_length_ft
FROM airports
WHERE runway_length_ft > 6000
```

Based on the combined information, associative attributes such as 'runway conditions' and 'weather' are derived. The associated information for the corresponding airports is retrieved from the database and then appended to the query answer, as shown in Figure 11. Since association can be transitive, we use Case Based Reasoning that based on past queries, user profile, and query context to terminate the association.

A Case Memory (see Figure 12) consists of *cases* and *association links*. Cases are past user queries (*e.g.*, Q_1, Q_2, Q_3, or Q_4). An association link (*e.g.*, l_1) is established by the attributes shared by the two cases (*e.g.*, Q_1 and Q_2), and the corresponding weight (*e.g.*, w_1) represents the usefulness of the association between the two cases. When a user query, Q_{user}, is executed, its conditions, user type, and query context are *compared* against the Case Memory for similar cases (*e.g.*, Q_1 and Q_2 are similar to Q_{user}). Based on the set of similar cases, a set of association subjects (Q_4 and Q_3) can be *selected* through the traversal of association links (l_2, l_3, l_4). The usefulness of an association is *computed* from the similarity measure of the corresponding cases and the weights of the traversed association links. The cases with the high usefulness values (*e.g.*, Q_3 and Q_4) are *adapted* into the user query (Q'_3 and Q'_4) as associations to the user. Initially, the Case Memory has not acquired any experience. *User feedback* on the usefulness of the associations is incrementally integrated into the Case Memory by adjusting the weights of the traversed association links. In this way, the Case Memory can accumulate experience from the user feedback.

Our current Case Base, consisting of about 1500 past queries, serves as the knowledge server for the association module. The size of the Case Base is around 2MB. For association purposes, we use the 300-case set which is composed of past queries used in the transportation domain. For testing performance and scalability of the system, we use a 1500-case set which consists of randomly generated queries based on user profile and query template over the transportation domain. Users can also browse and edit association control parameters such as the number of association subjects, associated links and weights of a given case, the threshold for association relevance, *etc.*
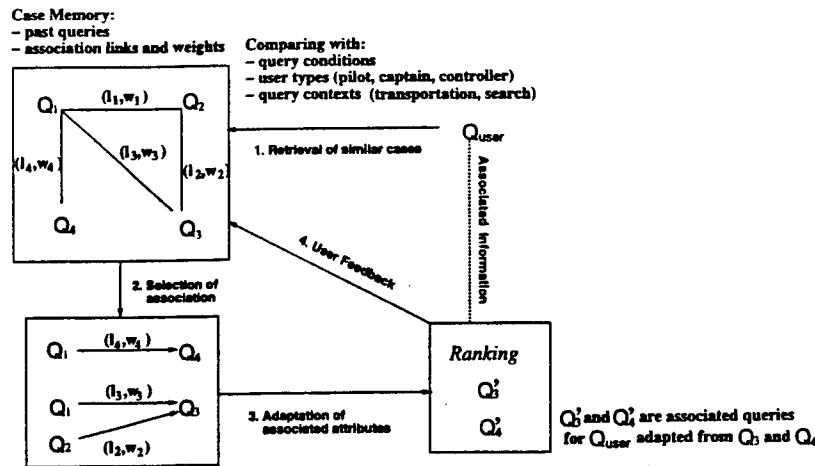
Figure 12. Case Memory and Associative Links.

5.5. Explanation Mediator

Query relaxation and association involve inference over vast data and knowledge sources. If users are to trust derived answers, it is vital to provide explanations of these processes. To not overwhelm the user, explanations should initially be summary. If users require further description, definition, or justification, they should be able to interactively obtain such explanations. Based on the user and context, explanations should occur at appropriate times during processing. One extreme is the system running automatically, only summarizing its work once it has completed. Another extreme is the system running in detail, explaining each action, giving the user the ability to monitor progress. Explanations should also be tailored to a user's understanding of the system. We have developed an explanation mediator (Minock and Chu, 1996) to serve these purposes.

The explanation mediator is *capable* of providing interactive, user-sensitive, and context-dependent explanations of CoBase cooperative operations. A client sends the explanation mediator a simple explanation request and receives back an explanation consisting of natural language and recommended visualizations. The client presents the explanation on their GUI and a simple protocol enables the user to interact with this initial explanation, causing the explanation mediator to generate follow up explanations. The explanation mediator *requires* access to queries, TAHs, execution traces, and answers (see Figure 13). CoBase mediators make this information available to the explanation mediator and the explanation mediator maintains a model of query processing, consisting of concepts and classification rules, by which it interprets such information. This approach is similar to the Explanation Explainable Expert System (EES) (Swartout et al., 1991) framework, though its model of query processing is built specifically to interpret CoBase operations, and work is focused on providing explanations any time during system execution. In addition, the representation, classification and generation formalism cover the complete explanation generation task, easing integration maintenance and extension.
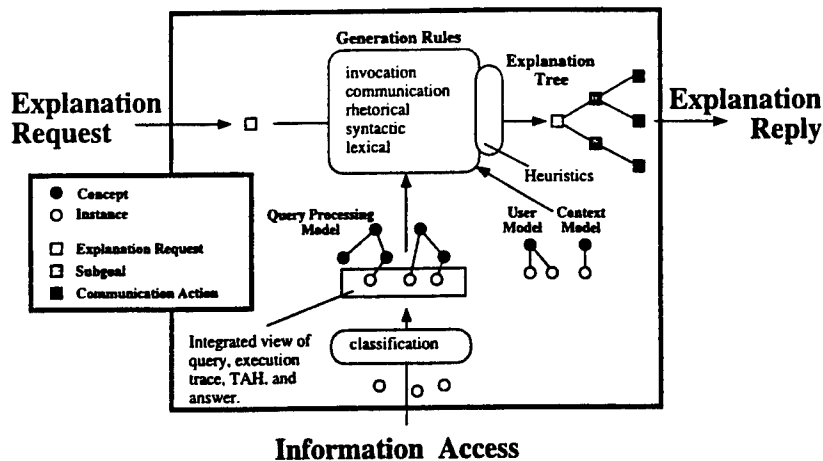
Figure 13. Explanation Mediator

The explanation mediator represents information such as queries, execution traces, TAHs, and answers in a representation similar to a semantic network. The particular representation, a directed hyper-graph, offers several advantages over standard semantic networks for explanation. Most importantly our hyper-graph allows for abstraction on the label linking nodes (instances, values, or concepts), enabling specific to generic access of the graph. The hyper-graph also enables the direct expression of one-to-many and many-to-many type relationships. Concepts in this graph (e.g. Query, Relax-Query-Action, TAH-node, etc.) include those of the query processing model. As the relaxation and association mediators execute, they send a stream of information that is instantiated in this graph. Classification rules in the query processing model interpret and augment this information as it added to the graph. In addition, the graph contains a user and context-model, enabling the generation of user-sensitive, context-dependent explanations.

The explanation mediator accepts an explanation request on a node (or set of nodes) in the graph and produces an explanation tree. This tree is created via the depth-first application of generation rules toward solving the initial explanation request. The explanation request is at the root of the explanation tree, while the text and recommended visualizations of the explanation are leaves in the tree. Intermediate nodes are the subgoals that were carried out to achieve the root explanation request. Generation rules consist of a goal, a set of constraints, and a set of actions. If the explanation request (or an intermediate subgoal) is matched by a generation rule goal, and if all of the rule's constraints are met, then the rule's actions are performed. These actions may either be subgoals that are in turn solved by application of generation rules or may be primitive communication actions (e.g. text or a visualization call).

A single inference mechanism applies classification and all levels of generation rules. Classification rules interpret CoBase information as instances of query processing concepts.

51

Generation rules expand explanation requests and intermediate nodes during the generation of an explanation tree. There are five types of generation rules: invocation, communication, rhetorical, syntactic and lexical. Invocation rules determine what should be explained under various user models and contexts. Communication rules determine what information will be expressed. Rhetorical rules determine the manners in which information is expressed. Syntactic rules constrain text to valid English. Lexical rules determine the names that will be given to CoBase actions and objects. Invocation and communication rules are typically the rules that match explanation requests while rhetorical, syntactic, and lexical rule typically expand intermediate nodes and produce primitive communication actions.

Different users require different types of explanation under different contexts. This is accounted for by the fact that more than one generation rule may match to expand a node in the explanation tree. Alternate generation rules vary in which aspects of CoBase's process or results should be expressed and to what depth. Heuristics control how an explanation is generated by controlling which rule among the matching rules will fire to expand a goal. Note that these heuristics apply only to generation rules which determine when and how to express information . The interpretation of the relaxation and association processes is non-heuristic.

The quality of explanations (with respect to completeness, correctness, and precision) depends on access to CoBase queries, execution traces, TAHs, and answers, and also on the effort expended in populating and refining the explanation rule-base. Elapsed times depend on access cost to such information and the computational complexity of synthesizing the explanation tree. The dominant cost is the synthesis of explanation trees, particularly the cost of matching generation rules to expand nodes. A RETE (Forgy, 1982) pattern-matcher performs these matches. Performance scales as the logarithm of the number of generation rules. The explanation mediator is implemented in C++ and CLIPS. There are approximately 60 concepts and 20 classification rules in the query processing model. There are approximately 80 general generation rules for explanation of CoBase cooperative operations. In two CoBase application domains, electronic warfare and transportation planning, we have extended these general generation rules to create 40 specific generation rules for the electronic warfare domain and 30 rules for the transportation planning domain. These domain specific generation rules were based on simple revisions of the general generation rules. The explanation mediator can usually produce explanations in under a second on a Sun SPARC 10 workstation.

Current work is focused on providing a method by which system administrators and users may interactively refine and extend explanations. Currently, programmers must write new generation rules. Instead, administrators, through a sequence of editing operations, should be capable of extending initially general CoBase explanations into precise, domain specific forms. By doing so, the extended generation rule-base is populated with domain specific rules. Through feedback and interaction, users should be able to refine explanations by altering the heuristic knowledge that determines which rules are selected.

5.6. Data Source Mediator

The Data Source Mediator (DSM) provides two services (Figure 14):

52

```
        peer mediators              Capability:
 ┌──────────┴────────────────┐         provide a uniform representation of data sources
 │    ┌───────────────────┐  │         convert query objects into queries
 │    │ Query and Schema  │  │
 │    │ Translation       │  │      Requirement:
 │    └───────────────────┘  │         query objects
 │  ┌──────┐ ┌──────┐ ┌───────┐        data sources and their schema
 │  │RDB   │ │Oracle│ │Objectivity│
 │  │Interface│ │Interface│ │Interface│
 │  └──────┘ └──────┘ └───────┘
 └────┼────────┼─────────┼────┘
   RDB      Oracle    Objectivity
   Database  Database  Database
```
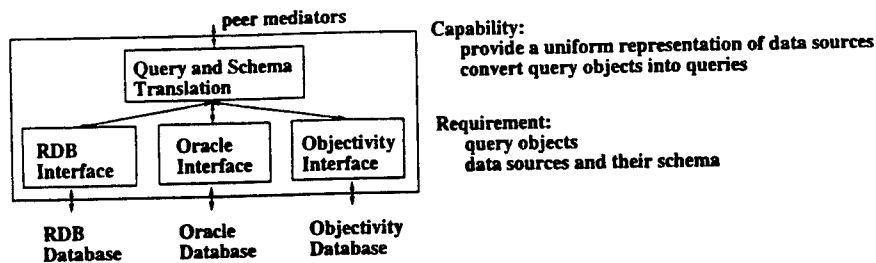
Figure 14. The Data Source Mediator

1. *Virtual Database Interface.* Different data sources usually have different schema. To extend CoBase to different data sources, a virtual database interface is developed. The virtual database interface represents a data source as a list of tables (each table consists of a list of attributes). DSM extracts schema information from a specific data source and builds the virtual database interface. As a result, data schema and data can be accessed from data sources without the need of knowing the differences of the underlying data sources. The *Query and Schema Translation* module converts a virtual database information request into a specific request supported by one of its underlying data sources.

2. *Query Conversion.* Different data sources may also use different query languages. When a query is processed in CoBase, it uses an internal representation called *query object*. To pose a processed query to a specific data source, DSM must convert query objects into the specific query language for the target data source (*e.g.*, SQL).

In DSM, each data source has a database interface to handle the specific requirements. Currently, we have database interfaces for Oracle (a relational database), RDB (a UNIX-based text database), and Objectivity (an object-oriented database). New data sources can be incorporated by adding a new interface module to the DSM, which does not affect the rest of the system. Thus, DSM makes CoBase scalable and extensible to different types of data sources.

5.7. Inter-Mediator Communications

Since CoBase mediators are independent processes possibly running on different computers, a communication language is needed to facilitate information exchange and service request among these mediators. Each CoBase mediator (Figure 15) consists of a *CoBase module* (*e.g.*, A or B), the *CoBase Ontology Layer* (for message interpretation), the *CoBase Content Language Layer* (for message representation), and *the KQML Layer* (for message transportation):

1. A CoBase module performs a specific cooperative service, such as relaxation, association, explanation, or TAH management. Each module has a set of *APIs* and represents its information in C++ *objects*. The service of a module is invoked through its APIs.
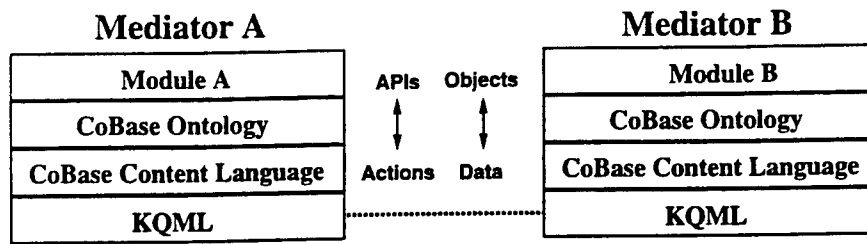
53

Module A	APIs Objects	Module B
CoBase Ontology	↑ ↑	CoBase Ontology
CoBase Content Language	Actions Data	CoBase Content Language
KQML	····························	KQML

Figure 15. Flow of mediator communications.

Since CoBase mediators are running in separated process spaces, an API call must be transported to the module supporting it. When APIs or objects are represented in a transportable format, they are called *CoBase messages*.

2. A CoBase Ontology (CO) provides basic vocabulary to compose CoBase messages for mediator communications. In a CoBase message, an API call is represented as an *action* and an object is represented as a *data*. For example, an API call for relaxing a given condition:

```
(generalize_condition TAH_NODE *tah_node)
```

is represented in CoBase Content language as

```
(generalize_condition "TAH_NODE_REFERENCE")
```

CO functions as a dictionary for the translation between APIs/objects and actions/data.

3. A CoBase Content Language (CCL) is used to represent CoBase messages in a structured form so that they can be interpreted and understood among different mediators. CoBase mediators require interactive communication capabilities. For example, an explanation returned by the explanation mediator contains some actions for further interaction. A user can select on certain phrases of the explanation to require the explanation mediator for further clarification, elaboration, or justification. Thus, data exchanged among CoBase mediators not only contain necessary information for the completion of an action, but also have *embedded* actions for further interaction. CCL is designed to support such capability.

4. The Knowledge and Query Manipulation Language (KQML) (Finin et al., 1993) serves as envelopes to pass CoBase messages among the CoBase mediators across the network.

Among the three components of the Communication Language, the Ontology (*i.e.*, CO) is developed based on CoBase modules and their APIs, while the content language (*i.e.*, CCL) and the transport language (*i.e.*, KQML) are CoBase-independent.

The flow of CoBase mediator communication is illustrated in Figure 15. The communication starts with a request for action from a mediator (*e.g.*, A). The request (*i.e.*, an API call) and its parameters (*i.e.*, C++ objects) are translated into action and data in CCL by

CO, wrapped inside a KQML message, and passed to a designated mediator (*e.g.*, B). After the message is received, it is unwrapped at the KQML Layer and translated back to an API call by CO. The call is performed by B and an answer is returned. The answer (in C++ object format) is translated into CCL and sent back to A, in the same manner as as it does from A to B.

The following example describes the use of CoBase messages for communication between the Relaxation Mediator (RM) and the TAH Manager (TM). TM provides TAH traversing operations such as generalization and specification. When RM needs to relax a condition via a TAH node with identifier "TAH_NODE_ID_113", it sends a request (*i.e.*, an API call) to TM:

```
(generalize_condition TAH_NODE_ID_113)
```

This call is translated into a CCL action:

```
(generalize_condition "TAH_NODE_ID_113")
```

This message is wrapped inside a KQML performative and passed to TM:

```
(achieve :language CoBaseContentLanguage
         :ontology CoBaseOntology
         :reply-with m_17
         :sender RelaxationMediator
         :receiver TahManager
         :content (generalize_condition "TAH_NODE_ID_113"))
```

TM returns relaxed conditions to RM:

```
"'BIZERTE' 'TUNIS' 'DJEDEIDA'"
```

Another example to illustrate the communication between RM and the Explanation Mediator (EM): a user requires the description of a relaxation process identified by "relaxQueryActioı 01".

```
(explain "relaxQueryAction-01" "describe")
```

EM returns an explanation to RM: "**The query** retrieving **airport name** where **runway length** is greater than 2000 feet, ... after relaxation **location name Bizerte** relaxed to **location names Tunis, Djedeida, ...**" (phrases in **bold** can be selected on for further explanation). The corresponding CCL is:

```
(DATA Explanation
      ("The query" :action (explain "node-ref-1")
       "retrieving"
       "airport name" :action (explain "node-ref-2")
       "where"          ..
       "runway length" :action (explain "node-ref-3")
       "is greater than 2000 feet"
       ...)
```

"node-ref-1," "node-ref-2," and "node-ref-3" are node identifiers in the explanation graph generated by EM for "relaxQueryAction-01." When a user selects a phrase which has a predefined action (*e.g.*, **The query, airport name**), the selected action is sent back to EM to perform the action and returns more explanation on the phrase (*i.e.*, **The query**, or **airport name**).

6. Implementation

6.1. Object-Oriented Implementation

We employ the object-oriented paradigm to implement CoBase. In this implementation, queries, database schema/interface, cooperative operators, and TAHs are represented as objects. Because of *polymorphism* and *encapsulation*, differences among various specific query languages, databases, cooperative functions, or types of TAHs are hidden abstract objects. Thus, query relaxation and other cooperative modules can be built upon these abstract objects with little knowledge about the implementation of the specific objects. For example, we have different objects for different databases including relational databases (*e.g.*, Oracle or SyBase object) and object-oriented databases (*e.g.*, Objectivity object). Each provides its own interface for database access. But CoBase modules only see a *database* object (i.e., a virtual database interface) which is an abstract interface to access any databases. This implementation enables these modules to work easily with different databases. Moreover, the object-oriented approach provides easy enhancements to CoBase. Adding a new cooperative function, for instance, simply entails defining a new cooperative operator object. The modification effects are localized and minimized through abstraction. Further, *inheritance* allows code sharing and reusing, thus reducing the development and maintenance effort. For example, functions common to all databases can be implemented in the abstract database object and inherited by all specific database objects. In this object-oriented paradigm, CoBase consists of multiple reusable modules. These modules are built on top of abstract objects. Thus, they can be easily ported to other systems by building *host-compliant* objects.

6.2. Graphical User Interfaces

CoBase's control mechanisms include graphical user interfaces for the Relaxation Mediator, Associative Mediator, and the Explanation Mediator. The GUI for the relaxation was implemented in C++ and X/MOTIF. It enables users to browse the database, visualize the query relaxation process, and view the answers. Further, a map server was integrated into CoBase. It allows a user to pose queries based on geographic objects such as countries, states, cities, airports, rivers, *etc.* An important feature of these objects is that they can be *spatially* related via 'located,' 'within,' 'contain,' 'intersect,' 'union,' and 'difference.' These spatial operators can be used in our system to specify spatial conditions in spatial queries. For example, a query such as, "find all the airports in the region specified on the map" retrieves the airports in the user-specified region and displays the airport objects on the
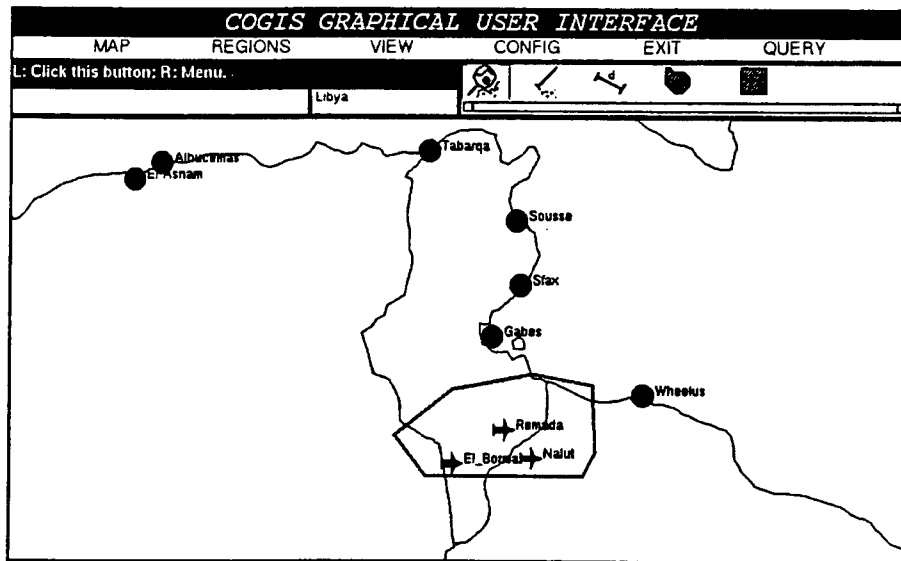
Figure 16. Find all the airports in a user-specified area.

map, as shown in Figure 16. More importantly, it allows a user to ask complex queries by the simple 'point-and-click' method. The GUI for the explanation supports a hypertext-like browser for explanation. It allows users to interactively ask for more definition, elaboration, justification, or simply summarization.

7. Performance Evaluation

In this section, we present the CoBase performance based on measuring the execution of a set of queries on the CoBase testbed developed at UCLA for the ARPI transportation domain. The performance measure includes response time for query relaxation, association, and explanation, and the quality of answers. The response time depends on the type of queries (*e.g.*, size of joins, number of joins) as well as the amount of relaxation, association, and explanation required to produce an answer. The quality of the answer depends on the amount of relaxation and association involved. The user is able to specify the relaxation and association control to reduce the response time and also to specify the requirement of answer accuracy. In the following, we shall show four example queries and their performances. The first query illustrates the relaxation cost. The second query shows the additional explanation cost, while the third query shows the additional association cost. The fourth query shows the processing cost for returned query answers as well as the quality of answers by using TAH vs MTAH for a very large database table (about 200,000 tuples).

Query 1: Find nearby airports can land C-5.

Based on the airplane location, the relaxation mediator translates 'nearby' to a pre-specified or user-specified latitude and longitude range. Based on the domain knowledge of C-5,

57

the mediator also translates 'land' into required runway length and width for landing the aircraft. The system executes the translated query. If no airport is found, the system relaxes the distance (by a predefined amount) until an answer is returned. In the above query, an airport is found after one relaxation. Thus, two database retrievals (*i.e.*, one for the original query, one for the relaxed query) are performed. Three tables are involved: Table GEOLOC (50,000 tuples), table RUNWAYS (10 tuples), table AIRCRAFT_AIRFIELD_CHARS (29 tuples). The query answers provide airport locations and their characteristics.

```
Elapsed time:   5 seconds processing time for relaxation
                40 seconds database retrieval time
```

Query 2: Find at least 3 airports similar-to Bizerte based on runway-length and runway-width.

The relaxation mediator retrieves runway characteristics of Bizerte airport and translates the similar-to condition into the corresponding query conditions (runway length and runway width). The system executes the translated query, and relaxes the runway length and runway width according to the TAHs until at least 3 answers are returned. Noted the TAH used for this query is a Runway-TAH based on *runway-length and runway-width*, which is different from the Location-TAH based on *latitude and longitude* (shown in Figure 8). The nearness measure is calculated based on weighted mean square error. The system computes similarity measure for each answer obtained, ranks the list of answers, and presents it to the user. The system obtains five answers after two relaxations. The best three are selected and presented to the user. Two tables are involved: table GEOLOC (50000 tuples), table RUNWAYS (10 tuples).

```
Elapsed time:   2 seconds processing time for relaxation
                10 seconds database retrieval time
```

The explanation mediator generates the following description of the SQL query:

> *The query retrieves airport name, runway length, and runway width where answers are similar to the runway length and runway width of the airport name Bizerte. A minimum of three answers are required.*

```
Elapsed time:   5 seconds processing time for explanation
```

The explanation mediator generates the following description of relaxation process:

> *Query relaxed: airport name Bizerte relaxed to Bizerte and El Borma.*

```
Elapsed time:   2 seconds processing time for explanation
```

> *Query relaxed: airport name Bizerte and El Borma relaxed to airport name Bizerte, Djedeida, El Borma, Gabes and Gafsa.*

```
Elapsed time:   2 seconds processing time for explanation
```

Further, the explanation provides the following summarization:

airport name Bizerte relaxed to Gafsa, Gabes, El Borma, Djedeida and Bizerte yielding 5 answers. These answers are ranked by similarity to the runway length and runway width of the airport name Bizerte based on weighted mean square error measure.

```
Elapsed time:   2 seconds processing time for explanation
```

Query 3: Find seaports in Tunisia with a refrigerated storage capacity of over 50 tons.
The relaxation mediator executes the query. The query is not relaxed, so one database retrieval is performed. Two tables are used: table SEAPORTS has 11 tuples; table GEOLOC has about 50000 tuples.

```
Elapsed time:   2 seconds processing time for relaxation
                5 seconds database retrieval time
```

The association mediator returns relevant information about the seaports. It compares the user query to previous similar cases and selects a set of attributes relevant to the query. Two top associated attributes are selected and appended to the query. CoBase executes the appended query and returns the answers to the user, together with the additional information. The two additional attributes associated are location name and availability of railroad facility near the seaports.

```
Elapsed time:   10 seconds for association computation time
```

Query 4: Find at least 100 cargos of code '3FKAK' with the given volume (length, width, height), code is non-relaxable.
The relaxation mediator executes the query, and relaxes the height, width, and length according to MTAH, until at least 100 answers are returned. The query is relaxed four times. Thus, five database retrievals are performed. Among the tables accessed, Table CARGO_DETAILS has 200,000 tuples, a very large table.

```
Elapsed time:   3 seconds processing time for relaxation using MTAH
                2 minutes database retrieval time for 5 retrievals
```

By using single TAHs (*i.e.*, single TAHs for height, width, and length respectively), the query is relaxed 12 times. Thus 13 database retrievals are performed.

```
Elapsed time:   4 seconds for relaxation by single TAHs
                5 minutes database retrieval time for 13 retrievals
```

For queries involving multiple attributes in the same relation, using an MTAH that covers multiple attributes would provide better relaxation control than using a combination of single-attribute TAHs. The MTAH compares favorably with multiple single-attribute TAHs in both quality and efficiency. We have shown that an MTAH yields a better relaxation strategy than multiple single-attribute TAHs. The primary reason is that MTAHs capture attribute-dependent relationships which cannot be captured when using multiple single-attribute TAHs.

Using MTAHs to control relaxation is more efficient than using multiple single-attribute TAHs. For the above example, relaxation using MTAHs require an average of 2.5 relaxation steps, while single-attribute TAHs require 8.4 steps. Since a database query is posed after each relaxation step, using MTAHs saves around 6 database accesses on average. Depending on the size of tables and joins involved, each database access may take from 1 second to about 30 seconds. As a result, using MTAHs to control relaxation saves a significant amount of user time.

With the aid of domain experts, the above queries can be answered by conventional databases. Such an approach takes a few minutes to a few hours. However, without the aid of the domain experts, it may take hours to days to answer these queries. CoBase incorporates domain knowledge as well as relaxation techniques. Thus, it is able to automatically search for the answer with significantly less time.

8. Technology Transfer of CoBase

CoBase stemmed from the transportation planning application for relaxing query conditions. CoBase has been linked with SIMS (Arens and Knoblock, 1992) and LIM (McKay et al., 1992) as a knowledge server for the planning system. SIMS performs query optimizations for distributed databases, and LIM provides high level language query input to the database. A Technical Integration Experiment (TIE) has been performed to demonstrate the feasibility of this integrated approach. CoBase technology is being implemented for the ARPI transportation and logistic planning applications. In addition, CoBase has also been successfully applied to the following domains.

In electronic warfare, one of the key problems is to identify and locate the emitter for radiated electro-magnetic energy based on the operating parameters of observed signals. The signal parameters are radio frequency, pulse repetition frequency, pulse duration, scan period, etc. In a noisy environment, these parameters often cannot be matched exactly within the emitter specifications. CoBase can be used to provide approximate matching of these emitter signals. A knowledge base (TAH) can be constructed from the parameter values of previously identified signals and also from the peak (typical, unique) parameter values. The TAH provides guidance on the parameter relaxation. The matched emitters from relaxation can be ranked according to relaxation errors. Our preliminary results have shown that CoBase can significantly improve emitter identification as compared to conventional database techniques, particularly in a noisy environment. From the line of bearing of the emitter signal, CoBase can locate the platform that generates the emitter signal by using the "near-to" relaxation operator.

In medical databases that store X-rays and MR images, the images are evolution and temporal-based. Further, these images need to be retrieved by object features or contents rather than patient ID (Chu et al., 1994). The queries asked are often conceptual and not precisely defined. We need to use knowledge about the application (e.g., age class, ethnic class, disease class, bone age etc.), user profile and query context to derive such queries (Chu et al., 1995). Further, to match the feature exactly is very difficult if not impossible. For example, if the query, "Find the treatment methods used for tumors *similar to* $X_i(location_{x_i}, size_{x_i})$ on 12-year-old Korean males," cannot be answered, then based on
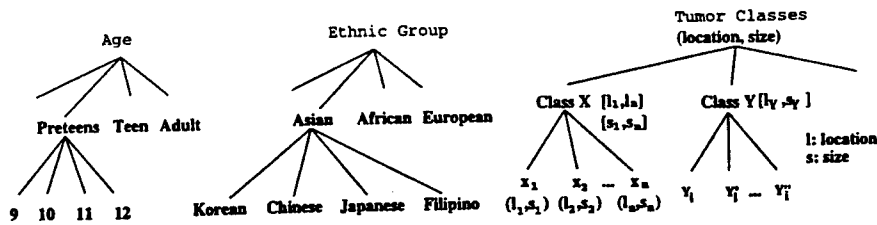
Figure 17. Type Abstraction Hierarchies for the Medical Query Example.

the TAH shown in Figure 17, we can relax "tumor X_i" to tumor "Class X," and "12-year-old Korean male" to "pre-teen Asian," which results in the following relaxed query: "Find the treatment methods used for tumor "Class X" on pre-teen Asians." Further, we can obtain such relevant information as the success rate, side effects, and cost of the treatment from the association operations. As a result, query relaxation and modification are essential to process these queries. We have applied CoBase technology to medical imaging databases (Huang and Taira, 1992). TAHs are generated automatically based on context-specific (*e.g.*, brain tumor) image features (*e.g.*, location, size, shape, *etc*). Once the TAHs for the medical image features have been constructed, query relaxation and modification can be carried out on the medical features.

The use of CoSQL constructs such as 'similar-to,' 'near-to,' and 'within' can be used in combination, thus greatly increasing the expressibility for relaxation. For example, we can express "find tumors *similar to* the tumor x *based-on* (shape, size, location) and *near to* object O *within* a specific range (*e.g.*, angle of coverage)." The relaxation control operators, such as matching tumor features in accordance to their importance, can be specified by *relaxation-order* (location, size, shape) to improve the relaxation quality.

9. Conclusions

We have presented a structured approach that uses Type Abstraction Hierarchy for providing query modification. TAHs provide multi-level knowledge representation and can be generated automatically from data sources. Further, TAHs are user and context sensitive, and easy to customize and maintain. Thus, TAHs are scalable to large systems. We have also extended the SQL to CoSQL. This extended language provides cooperative operators that allow the user to explicitly specify relaxation operations and controls. These operators are very general and can be applied to different domains. Combination of these operators in a query can greatly improve the expressive power. Based on user profile and application contexts, the relaxation manager limits the search scope and also filters out the unsuitable answers. To provide relevant information to the query answers, a pattern-based framework is used for deriving associative (relevant) information from past cases. An explanation facility is included which summarizes the query modifications and relaxation process, and also

provides the nearness of the approximate answer to the exact answer. Further, a cooperative geographic information system has been developed to provide information retrieval and analysis based on geographic objects (*e.g.*, countries, cities, airports) and spatial relations (*e.g.*, 'located,' 'within,' 'between,' 'near-to,' 'closest'), and to perform context-sensitive relaxation of geographic relations. CoBase is implemented in a mediator-based architecture which is scalable and extensible.

Performance measurements on our testbed reveal that the cost for relaxation, explanation, and association is fairly small. The major cost is due to database retrieval which depends on the number of relaxations required before obtaining a satisfactory answer. For queries with multiple dependent attributes, the use of MTAH can reduce the number of relaxations, thus decreasing the overall cost.

CoBase stemmed from the ARPA/Rome Lab Planning Initiative for providing approximate query answers from a large and heterogeneous database. It has also been applied in a medical imaging database (X-ray, MRI) for approximate matching of image features and contents, and in electronic warfare for approximate matching of emitter signals (based on a set of parameter values) and also for locating the platforms that generate the signals via spatial relaxation.

10. Acknowledgment

Notes

1. The definition of abstraction allows successive single-value abstractions, which does not provide real abstraction, therefore in implementation, they are collapsed into a single abstraction.
2. The goodness is defined by the intra-class similarity and inter-class dissimilarity.
3. A cut c is a value that separates a cluster of numbers $\{x | a \leq x \leq b\}$ into two sub-clusters $\{x | a \leq x \leq c\}$ and $\{x | c < x \leq b\}$.
4. Dependency here means that all the attributes as a whole define a coherent concept. For example, the length and width of a rectangle are said to be "semantically" dependent. This kind of dependency should be distinguished from the *functional dependency* in database theory.

References

Arens, Y. and C. Knoblock. Planning and reformulating queries for semantically–modelled multidatabase systems. In *Proceedings First International Conference on Information and Knowledge Management (CIKM)*, pages 92–101, Baltimore, Maryland, 1992.

Chu, Wesley. W. and Q. Chen. A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering*, 6(5):738–749, 1994.

Chu, Wesley W. and Kuorong Chiang. Abstraction of high level concepts from numerical values in databases. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, July 1994.

Chu, Wesley W., Qiming Chen, and Rei Chi Lee. Cooperative query answering via type abstraction hierarchy. In S.M. Deen, editor, *Cooperating Knowledge Based Systems*, pages 271–292. Springer-Verlag Inc., October 1991.

Chu, W.W.,I.T. Ieong, and R.K. Taira. A semantic modeling approach for image retrieval by content. *Journal of Very Large Database Systems*, 3:445–477, October 1994.

Chu, Wesley W., A.F. Cardenas, and R.K. Taira. KMeD: A knowledge-based multimedia medical distributed database system. *Information Systems*, 20(2):75–96, 1995.

Cuppens, F. and R. Demolombe. Cooperative answering: a methodology to provide intelligent access to databases. In *Proceedings of the 2th International Conference on Expert Database Systems*, Virginia, USA, 1988.

Cuppens, F. and R. Demolombe. How to recognize interesting topics to provide cooperative answering. *Information Systems*, 14(2):163–173, 1989.

Fouque, G., W.W. Chu, and H. Yau. A case-based reasoning approach for associative query answering. In *Proceedings of the Eighth International Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, October 1994.

Fisher, D. H. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

Forgy,C.L. Rete: A fast algorithm for the many pattern/many object pattern match problem, 1982.

Finin, T.W., J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, J. Pelavin, S. Shapiro, and C. Bech. Specification of the KQML agent-communication language, plus example agent policies and architectures (draft). prepared by the ARPA Knowledge Sharing Initiative External Interfaces Working Group, June 1993.

Gluck, M. A. and J. E. Corter. Information, uncertainty, and the unity of categories. In *Proceedings of the 7th Annual Conference of the Cognitive Science Society*, pages 283–287, Irvine, CA, 1985.

Gaasterland, T., P. Godfrey, and J. Minker. An overview of cooperative answering. *Journal of Intelligent Information Systems*, 1:123–157, 1992.

Gaasterland, T., P. Godfrey, and J. Minker. Relaxation as a platform of cooperative answering. *Journal of Intelligent Information Systems*, 1:293–321, 1992.

Hemerly, A.S., M.A. Casanova, and A.L. Furtado. Exploiting user models to avoid misconstruals. In R. Demolombe and T. Imielinski, editors, *Nonstandard Queries and Nonstandard Answers*, pages 73–98. Oxford Science Publications, 1994.

Hemerly, A.S., A.L. Furtado, and M.A. Casanova. Towards cooperativeness in geographic databases. In *Proc of the 4th International Conference on Database and Expert Systems Applications*, Prague, Czech Republic, 1993.

Huang, H.K. and R.K. Taira. Infrastructure design of a picture archiving and communication system. *American Journal of Roentgenology*, 158, 1992.

Merzbacher, Matthew and Wesley W. Chu. Pattern-based clustering for database attribute values. In *Proceedings of AAAI Workshop on Knowledge Discovery*, Washington, DC, 1993.

Minock, M. and W.W. Chu. Interactive explanation for cooperative information systems. Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems, Zakopane, Poland, 1996, Springer Verlay lecture notes for Artificial Intelligence.

Mark, D.M. and A.U. Frank. Concepts of space and spatial language. In *Proceedings of the 9th International Symposium on Computer-Assisted Cartography*, pages 538–556, Baltimore, MD, April 1989.

Motro, A. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, 1988.

McKay, D. P.,J. Pastor, and T. W. Finin. View-Concepts: Knowledge-based access to databases. In *Proceedings First International Conference on Information and Knowledge Management (CIKM)*, pages 84–91, Baltimore, Maryland, 1992.

Subramanian, R. and N.R. Adam. Ill-defined spatial operators in geographic databases: their nature and query processing strategies. In *Proceedings of ACM Workshop on Advances in Geographical Information System*, pages 88–93, Washington, DC, October 1993.

Swartout, W., C. Paris, and J. Moore. Design for explainable expert systems. *IEEE Expert*, 6(3):58–64, June 1991.

Wiederhold, G. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

Associative Query Answering via Query Feature Similarity*

Wesley W. Chu and Guogen Zhang
Computer Science Department
University of California, Los Angeles, CA 90095, USA

Abstract

Associative query answering provides additional relevant information to the queries that is not explicitly asked, but is of interest to the user. For a given query, associative information may be derived from past user query cases based on the user type and the query context. A case-based reasoning approach that matches query features is proposed. Query feature consists of the query topic, the output attribute list, and the selection constraints. The similarity of the query feature is defined and can be evaluated from the semantic model that is derived from the database schema. Query feature based associative attribute search is presented.

1 Introduction

Associative query answering [1, 2, 3] is to provide users with additional relevant information to the query that the user did not ask or does not know how to ask, and may be of interest to the users. For a given query, the associative information depends on the user's intention, interests, and query context. For example, for the query: *"List airports in Tunisia that can land a C-5 cargo plane,"* the relevant information for a transportation planner may include the *railway facility* at the airports. But for a pilot, the interested information may be the *runway condition* and *weather condition.*

Attribute based association from case base has been used [3]. However, the derived associative information often is not domain and user specific. To remedy such shortcoming, we propose to use query feature that includes query topic as additional information to provide relevant information. As a result, the proposed approach yields more relevant information than that of the attribute based search approach [3], which is only based on the query output attribute list and selection conditions. Further, the domain knowledge from a semantic model is used to evaluate query feature similarity to improve the efficiency in searching associative information from case bases.

The paper is organized as follows. In section 2, we discuss our approach and present an overview of the associative query answering system. In section 3 we describe a semantic model based on the database schema used for topic identification and query similarity evaluation. Then in section 4 query feature and query similarity are discussed. The search and learning process is described in section 5.

2 Associative query answering systems
2.1 Approach

Due to the unlimited number of user queries possible for a relational database, it will be very costly to derive associative information by knowledge engineers [4]. Therefore, a case-based reasoning approach [3] is used to capture the domain and user-specific information for associative query answering.

Given a set of past queries, searching for associative attributes can be based on the frequencies of attribute groups that appeared in the queries. However, such approach does not provide context of co-occurrent attributes. Therefore we propose to use the information in the database schema and the query to analyze the user intention to guide query association.

For an SQL query, three types of associative attributes and expressions are identified: 1) *simple associative attributes*: attributes of relations in the current query; 2) *extended associative attributes*: attributes of relations introduced into the query by joins with the existing relations; and 3) *statistical associative information*: aggregate functions, e.g. count and sum, that are related to the main entity in the query. In this paper, we only focus on the first two types.

2.2 System overview

The associative query answering process is illustrated in Fig. 1. A semantic model is constructed from the database schema and domain knowledge. Each user can derive his own domain-specific semantic model from the general semantic model via user-specific knowledge. A case base stores past queries and the corresponding association cases. After an SQL query converts to a query feature vector, the case base is searched based on the similarity of the input query to queries in the case base. The matched associative attribute candidates are displayed to user for selection.
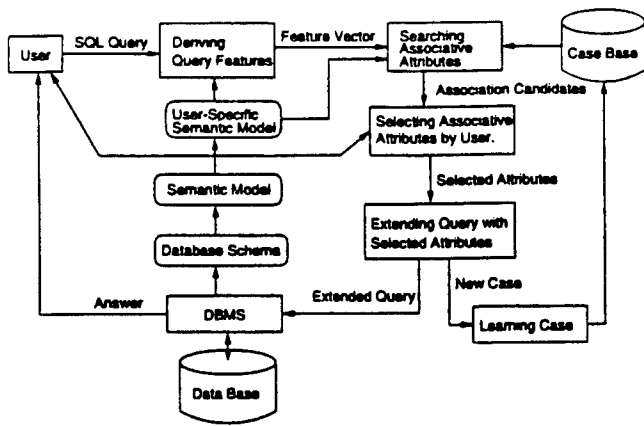
Figure 1: The associative query answering process by query feature matching.

The query is extended with the selected attributes and is sent to the DBMS for execution. Each query, with its association and user feedback, is recorded in the case base for future reasoning use.

For a given query, the system first locates similar query cases with the same topic. If such query cases exist, the associative attributes are used as the association. Otherwise, query cases with related topics are searched and the query feature similarity is used to guide the search process.

3 Database schema and semantic model

In relational databases, relations are constructs used for representing entities and relationships [5]. To explicitly express the semantics in the relations, we construct a semantic (EER) model based on the database schema [6], user-defined relationships, and mapping from its constructs into the relations in the database.

The semantic model is represented as a directed graph called semantic graph. The nodes in the graph are entities and complex relationships that have corresponding relations in the database. The edges are links (i.e. joins) among relations. The user can define additional nodes and edges for his views in his own model. The semantic graph is further divided into a set of overlapping query contexts, each of which is a connected component of the user's semantic graph, and reflect the user's interests on the database when performing a certain task. A query topic is a connected component of a query context. A query is a constraint on a topic, i.e. a selection on instances of a topic, with an output attribute list. Fig. 2 shows a context of the semantic model about aircrafts and airports for a transportation database, where "landing"
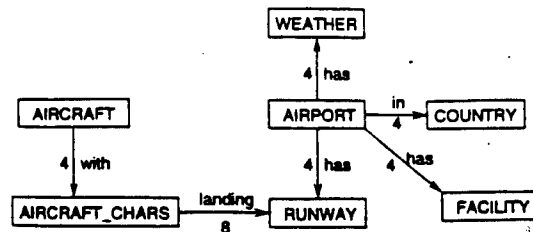

Figure 2: A query context about AIRCRAFT and AIRPORT for a transportation database.

is a user-defined link.

To compare the similarity of query topics, we assign weights for the nodes and edges in each query context. Weights are used to measure the relative importance of the semantic contents of the links in the graph. An equal unity weight is assigned to all nodes. Weights for the edges are based on their semantic contents, or the context specificity of the links. For example, in Fig. 2 "landing" is a more specific link compared to other links, thus a larger weight is assigned. The weight for a path is the sum of weights of the nodes and edges on the path. For any two nodes, if one path connecting them has larger weight than another, the larger one is considered containing more specific semantics than the smaller one [7, 8]. Let queries Q_1 and Q_2 contain a common path with weight w_a, Q_1 and Q_3 also have a common path with weight w_b. If $w_a > w_b$, then the topic of Q_1 is more similar to Q_2 than Q_3. The same links may have different relative importance under different contexts, and the weight assignment is user and context sensitive.

4 Query feature and similarity
4.1 Query feature vector

A query can be characterized with the following components:

1. a topic graph, T, defining the topic of the query;

2. an output attribute list, O, defining the queried aspect on the topic;

3. a set of selection conditions, C.

We shall denote $Q = \langle T, O, C \rangle$ as the *feature vector* of a query.

For an SQL query with a single SELECT statement, a feature vector can be constructed. Fig. 3 shows the SQL query and its feature vector for the example stated in the introduction.

4.2 Topic relationships among queries

For two queries $Q_1 = \langle T_1, O_1, C_1 \rangle$ and $Q_2 = \langle T_2, O_2, C_2 \rangle$, the relationships of their topics can be used to derive the associative information.
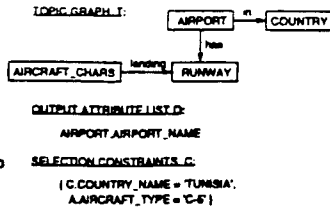
Figure 3: Feature vector for the example query.

1. *Same topic*: $T_1 = T_2$. If $O_2 - O_1 \neq \emptyset$, then $O_2 - O_1$ may be used as associative attributes for Q_1.

2. *Sub-topic*: $T_1 \subset T_2$. T_1 is a subtopic of T_2, and T_2 is a super topic of T_1. T_1 can be extended to T_2 and $O_2 - O_1$ can be used as possible associative attributes for Q_1.

3. *Overlapped topics*: $T_1 \not\subset T_2$ and $T_1 \not\supset T_2$, but $T_1 \cap T_2 \neq \emptyset$. Similar to the sub-topic relationship, $T_2 - T_1$ can be used to extend T_1, and $O_2 - O_1$ are possible associative attributes for Q_1.

4. *Unrelated topics*: $T_1 \cap T_2 = \emptyset$. The topics of Q_1 and Q_2 are not directly related. No associative information can be derived from an unrelated query.

4.3 Query similarity

Query similarity is based on three components of the feature vector: query topic, output attribute list, and constraints. Their relative importance is given by their corresponding weights: w_t, w_o, and w_c, with $w_t + w_o + w_c = 1$. The weights are user and context dependent.

For an input query $Q_1 = \langle T_1, O_1, C_1 \rangle$ and a query in the case base $Q_2 = \langle T_2, O_2, C_2 \rangle$, the similarity of Q_1 and Q_2 is defined as:

$$
\begin{aligned}
\theta(Q_1, Q_2) = \quad & w_t \times \theta_t(Q_1, Q_2) + \\
& w_o \times \theta_o(Q_1, Q_2) + \\
& w_c \times \theta_c(Q_1, Q_2)
\end{aligned}
\tag{1}
$$

where $\theta_t()$, $\theta_o()$, and $\theta_c()$ are the similarity functions for the topics, output attribute lists, and selection constraints, respectively.

The *topic similarity* is based on the size of the common topic components relative to the size of both topics together as follows:

$$
\theta_t(Q_1, Q_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}
\tag{2}
$$

where $|T|$ is the size of the topic, defined as the sum of weights of all nodes and edges. For instance, if the two topics are the same, then their topic similarity is 1. If a topic is a sub-topic of another, their similarity

is the ratio of the size of the sub-topic to that of the super topic. Non-related topics have similarity of 0.

The *similarity of output attribute lists* is defined as:

$$
\theta_o(Q_1, Q_2) = \frac{|O_1 \cap O_2|}{|O_1|}
\tag{3}
$$

which is based on the number of attributes in O_1 contained in O_2, and reflects the proximity of the queried aspect of Q_2 to that of Q_1.

The *similarity of selection constraints* is based on two factors: the number of the identical condition patterns and the similarity of values for these condition patterns. To evaluate the similarity, we divide C into a mapping S and a tuple of constants V. When the mapping is supplied with the constant values as the arguments, the set of selection conditions is generated. For example, if we have two selection conditions:

$$C.COUNTRY\_NAME= \text{'TUNISIA'},$$
$$A.AIRCRAFT\_TYPE = \text{'C-5'}$$

then we have $S = \{C.COUNTRY\_NAME = \$1, A.AIRCRAFT\_TYPE = \$2\}$, and $V = (\text{'TUNISIA',' DC}-8-61')$, where \$1 and \$2 refer to corresponding components of the tuple. $S(V)$ is the above selection conditions. We have

$$
\theta_c(Q_1, Q_2) = \theta_s(Q_1, Q_2) \times \theta_v(Q_1, Q_2)
\tag{4}
$$

where $\theta_s(Q_1, Q_2)$ is the similarity of mappings, defined as:

$$
\theta_s(Q_1, Q_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}
\tag{5}
$$

and $\theta_v(Q_1, Q_2)$ is the similarity of constraints values for the common conditions. Assume there are k common conditions and the projections of tuples V_1 and V_2 onto these common conditions are represented as sets $D_{i1}, \ldots, D_{ik}$ for $i = 1, 2$, $\theta_v(Q_1, Q_2)$ is defined as:

$$
\theta_v(Q_1, Q_2) = \frac{1}{k} \times \sum_{j=1}^{k} (1 - RE(D_{1j}, D_{2j})).
\tag{6}
$$

where $RE()$ is the relaxation error [9] for two sets of values, which is the average pair-wise distance of elements from each set. The $RE()$ values are provided from a tree-structured classification hierarchy TAH [10, 11] and is normalized into $[0, 1]$. As exceptions to equation 4, if $|S_1 \cup S_2| = 0$ (i.e. no selection conditions in both queries), then $\theta_c(Q_1, Q_2) = 1$. And if one of them is empty, say S_1, then the S_1 is treated as equal to S_2, with constants being the projections of the attributes from the database.
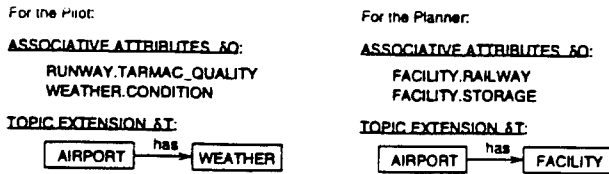
Figure 4: Possible associations for the example query.

5 Searching associative attributes from case base

Given an input query $Q = \langle T, O, C \rangle$, an extended query with associative information is $Q' = \langle T \cup \delta T, O \cup \delta O, C \rangle$, where δO is a set of the associative attribute candidates and δT is the necessary topic extension to derive these associative attributes. The associative extension denoted as $\langle \delta O, \delta T \rangle$ should not change the user's constraints on the query topic. Our goal is to determine $\langle \delta O, \delta T \rangle$ for the query Q. Fig. 4 shows the associations for two user types for the query example, and the corresponding query context and feature vector are shown in Fig. 2 and 3.

5.1 Search process

Searching of case base is done for each user type and context. Each query case in the case base is represented by its feature vector, and a log of past associative extensions of form $\langle \delta O, \delta T \rangle$ with positive user feedback $\langle \delta O_p, \delta T_p \rangle$ when each time the case is applied, where $\delta O_p \subseteq \delta O$ and $\delta T_p \subseteq \delta T$ and they are user selected attributes and corresponding topic extension. Thus, $\delta O - \delta O_p$ and $\delta T - \delta T_p$ are the negative feedback. Each case is time-stamped for ordering purposes.

The case base is organized based on the query topics at the first level, the output attribute lists at the second level, and then selection constraints at the third level. The cases are classified into groups based on these levels. The relationships among topics are linked to facilitate the search.

Given an input query $Q = \langle T, O, C \rangle$, the search process is outlined as follows:

1. If there is a set of cases with the same topic as T, use the *same topic matching*, in which search for associative attributes is limited within this case group. If the same topic matching is successful, the search is completed;

2. Otherwise, use *related topic matching*, where search is first expanded into the sub-topics and super topics, and then to the overlapped topics if necessary.

The success of the search process is based on the number of relevant associative attributes required, which

can be determined either by system default heuristics or by the user.

The associative candidates are displayed to the user for selection. The selected associative attributes (positive feedbacks) are extended to form a new query. The associative extension is recorded into the case base. The details of the search will be discussed in the following sections.

5.2 Same topic matching

The set of query cases with the same topic as the input query are searched. The single most similar query case is used at first and its log is retrieved for evaluation. The search process terminates if the associative attributes provide sufficient relevant information. Otherwise, less similar query cases are considered for association, and the process is repeated until sufficient relevant associative information is obtained or all the cases with the same topic are exhausted.

The usefulness evaluation for an attribute is based on the similarity of the query and usefulness weight of the attribute, which is calculated from the feedback history. Two different schemes based on history importance are proposed for usefulness weight calculation. One is to give all the cases in the history the same importance, which we call *equal importance* (EI) scheme. The other, called *recently used* (RU) scheme places more importance on recent cases. The former is suitable for a stable database and users with stable behavior, and the latter more suitable for a database incurring many updates or users with transient behaviors.

For the EI scheme, the weight assigned to an associative attribute can be calculated as follows:

$$W_{EI} = \frac{1 + N_p}{d + N_p + N_n} \qquad (7)$$

where N_p and N_n are the positive and negative counters respectively, and $d \geq 1$ is used to control the effect of the first few feedbacks.

For the RU scheme, all the feedbacks are ordered based on the time sequence, and N_p, N_n, R_p, and R_n are the total positive and total negative feedbacks, and recent positive and negative feedbacks, respectively. The recent feedbacks are counted based on the most recent turning points in the feedback sequence. The weight is calculated as:

$$W_{RU} = (1 - \alpha) \times \frac{N_p}{N_p + N_n} + \alpha \times \frac{R_p}{R_p + R_n} \qquad (8)$$

where $0.5 < \alpha \leq 1$ is used to control the relative significance of the recent feedbacks.

The relevancy of an associative attribute from the same query cases is the usefulness weight of the attribute from equation 7 or 8, times the similarity of

the queries from equation 1. The total usefulness is the weighted average from each group by the number of cases. If the total usefulness is greater than a certain threshold (e.g. 0.3), then it is a useful associative attribute. The threshold value depends on context, user profile, and should be estimated experimentally.

5.3 Related topic matching

If the association search on the same topic query cases does not provide sufficient information, then search is expanded to the sub-topic and super topic groups, and to the overlapped topics.

For the set of queries $\{Q_i = \langle T_i, O_i, C_i \rangle | i = 1, \ldots, n\}$, where n is the number of cases under evaluation, each attribute in O_i is assigned a weight in accordance with query similarity (equation 1). All the attributes from the searched queries are merged. The weight for an attribute $o \in \cup_i O_i$, $W(o)$, is:

$$W(o) = \sum_{i=1}^{n} m(o, O_i) * \theta(Q, Q_i), \qquad (9)$$

where the membership function $m(o, O_i)$ is 1 if $o \in O_i$, and 0 otherwise. The list is sorted based on the attribute weights.

For query cases with super topics, both simple associative attributes and extended associative attributes can appear in the result. For each extended attribute, the corresponding topic is used for deriving δT, which is the path in the super topic from the node containing the attribute to a node in the input query topic graph.

6 Conclusions

A query feature based associative query answering from case base is proposed in the paper. A query feature consists of its topic, its output attribute list, and the selection constraints. For a given query, the topic similarity of cases is evaluated from a user-specific semantic model based on the database schema, user type, and context. A case base that records the past user queries and association cases is used for searching for associative attributes, which is based on the user type, context, and the query features. Cases with a same topic are searched first, then followed by searching the related cases with sub-topics, super topics, or overlapped topics if the same topic searching does not provide any relevant information. We are currently in the process of implementing the proposed approach on the CoBase system [9]. We plan to evaluate the improvement of using the proposed query feature approach as compared with the attribute linking approach which searches past cases that have a same attribute either from the output attribute list or the selection constraints [3].

References

[1] F. Cuppens and R. Demolombe, Extending answer to neighbor entities in a cooperative answering context. *Decision Support System*, pp. 1-11, 1991.

[2] W. W. Chu and Q. Chen, Neighborhood and Associative Query Answering. *Intelligent Information Systems*, 1(3-4):355-382, 1992.

[3] G. Fouqué, W. W. Chu, and H. Yau, A Case-Based Reasoning Approach for Associative Query Answering, in *Proc. of the 8th SMIS*, Oct., 1994, NC.

[4] J. F. Allen and C. R. Perrault, Analyzing Intention in Utterances, *Artificial Intelligence*, 15: 143-178, 1980.

[5] J. D. Ullman, *Principles of database and knowledge-base systems*, Vol.II, Computer Science Press, 1988.

[6] R. H. L. Chiang, T. M. Barron, and V. C. Storey, Reverse engineering of relational databases: Extraction of an EER model from a relational database, *Data & Knowledge Engineering*, 12:107-142, 1994.

[7] Y. E. Ioannidis and Y. Lashkari, Incomplete path expressions and their disambiguation, SIGMOD'94, pp.138-149.

[8] J. A. Wald and P. G. Sorenson, Resolving the query inference problem using Steiner trees, *ACM Transactions on Database Systems*, Vol.9, No. 3, pp. 348-368, Sept. 1984.

[9] W. W. Chu, H. Yang, K. Chiang, M. Minock, G. Chow, C. Larson, CoBase: A Scalable and Extensible Cooperative Information System, *Journal of Intelligent Information Systems*, 1996.

[10] W. W. Chu and K. Chiang, Abstraction of high level concepts from numerical values in databases. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Seattle, 1994.

[11] M. Merzbacher and W. W. Chu, Pattern-based clustering for database attribute values. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Washington, D.C., 1993.

Associations and Roles in Object-Oriented Modeling*

Wesley W. Chu and Guogen Zhang
{wwc, gzhang}@cs.ucla.edu

Computer Science Department
University of California, Los Angeles, CA 90095

Abstract. We present an extended ER model with entity, role, and association as the basic constructs for object-oriented modeling. The purpose of the constructs is to support object evolution and extension for long lived objects. A class hierarchy consists of a static part and a dynamic part. The static part is a classification of entity classes, while the dynamic part is the role classes played by entities. The interaction among objects are captured with association classes. Based on the observation that entities play roles in association with other entities, we provide a unified view on roles in associations and roles as an extension to objects. The proposed modeling constructs help developers better understand the interrelationship among entities, thus result in flexible implementations for dynamic systems.

1 Introduction

Object-oriented systems provide more constructs than relational systems to model and support application semantics. However, in the current object-oriented models there is a lack of direct association support and a lack of systematic support for dynamic object evolution. In this paper, we introduce an entity-role-association modeling framework. *Entity* classes will be used to capture the static ISA hierarchy of the fundamental objects being modeled in the system. *Role* classes will be used to capture the ISA hierarchy of roles that entities can dynamically assume throughout their lifetime in the system. Further, the *association* class construct is introduced to explicitly model the relationships entities can participate in as they evolve through various roles during their lifetime. Altogether, our entity-role-association model provides a useful extension to OO modeling that supports dynamic object evolution. In our model, all the instances of entity classes, roles classes, and association classes are objects.

Relationships among entities is one of the fundamental constructs in semantic modeling as indicated in the original Entity-Relationship (ER) model [Chen76] and later extensions [TYF86, HK87, PM88]. The relationships in the original ER can be viewed as refined into different relationships in later extensions, such

as ISA relationship and others. *Association* is the term commonly used to refer to the interaction among objects [Rat97]. The associated objects are the *roles* of the association. In object-oriented systems, objects interact with each other by sending and reacting to messages. Real world entities are usually modeled by objects, and the state and other properties of associations often have to be encapsulated into the role objects involved. However, the properties of interactions, or associations, can be captured and represented in separate "association objects".

Tanzer [Tan95] presented a critique on the current practice of using regular instance variables for associations. The main problems of using regular classes and pointer/reference instance variables for associations are: (1) For classes with existing objects, it is not easy to insert an instance variable for a new association. (2) Constructing association instances using regular classes for polymorphic associations needs typecase testing on the roles. (3) Using direct multiple dispatching [Ing86] or the visitor pattern [GHJV95] to achieve multiple polymorphism for associations is a complex scheme. Further, it requires the methods to be added into role classes, thus increases code dependencies among modules, and is only partially extensible. In this paper, we propose *association class* construct for complex association modeling and support, and provide implementation schemes to resolve these issues.

We have used the role concept within the context of associations above. Let us now introduce another role concept in *object with roles* for dynamic aspects of objects [GSR96, RS91, Pern90, Alba93, WCL97]. In traditional class-based object-oriented systems, an object is uniquely represented by an instance of the most specific class in the class hierarchy that the object qualifies. Sometimes the static ISA relationship results in complex class hierarchies, and causes problems for object evolution. *Object with roles* is a way to remedy these problems. Under this extension, an object has an instance corresponding to the base (role) class, and a set of role objects that the object is currently playing. An object can acquire roles or drop roles dynamically. A role implicitly inherits the properties of its player. Object with roles extension provides object-oriented systems with more flexibility and expressive power.

For example, a person in a university can be a student. He can also be an employee if he is a TA or RA. The traditional class hierarchy is shown in Fig. 1a. If John was only a student at the beginning, he was represented by an instance of "Student" class. But later he became a TA, and had to be represented with an instance of "Student-Employee" class, while keeping the same object identity, which is a problem. An instance of "Student" cannot become an instance of "Student-Employee" without change of its object identifier. If we create a new instance, all the references to the old object will have to be changed to the new instance, which is very costly if not impossible[2]. If both instances are kept, there will be some redundancy, causing potential inconsistency. We call this *object reclassification anomaly*. Using objects with roles, "Person" can be an entity class, and "Student" and "Employee" are two role classes of "Person"

---

[2] Some systems use **becomes** to support this. But a new object needs to be created.

(Fig. 1b). Thus John can be represented by a `Person` object with a `Student` role at first. He acquired the employee (TA) role when he became a TA later. Multiple inheritance is also avoided in this case.

(a) ISA Hierarchy   (b) Class-Role Hierarchy

☐ Entity   ⟶ ISA   ⌐⎯⌐ Role   - - -▷ RoleOf
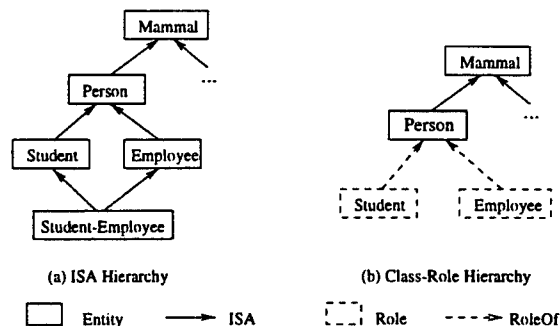
Fig. 1. ISA hierarchy and Class-Role hierarchy

Current research on *object with roles* and associations does not connect these concepts together. Based on the observation that an entity can play a role in association with other entities, we propose to unify the role concepts in *object with roles* and in *associations*. Thus roles can be unified both conceptually and in implementation under the entity-role-association scheme.

The paper is organized as follows. In section 2, we give an overview on modeling using entity-role-association. In section 3, we discuss association classes and their features. In section 4, we discuss the relationships of associations related to roles. Then in section 5, we discuss the relationships between players and roles. An application example is presented in section 6. And finally, a comparison of other related work is presented in section 7.

2 Modeling with Entities, Roles, and Associations

In our approach, the ISA relationship is subdivided into subclassing and role-playing, the entity classes constitute a static classification hierarchy while the role classes are dynamic aspects the entities can assume. The roles are played by objects, which can be entities or other roles.

The entity classes are those that reflect the static aspects of the real-world objects. The relevant entity classes are classified into a class hierarchy, in which classes have the partial order ISA relationship.

Role classes capture the temporal and evolutionary aspects of the real-world objects. The role classes themselves may also have class hierarchies to factorize the common properties with the ISA relationship. The semantics of ISA relationship for role classes are the same as the regular class inheritance (subclassing). For example, `Undergraduate` ISA `Student`, and `Graduate` student ISA `Student`. The player classes and role classes are related through RoleOf relationships, e.g.

Student is RoleOf Person, ProjectManager is RoleOf Employee as shown in Fig. 2.

Objects assume roles in their associations with other objects. Associations may have attributes for states and a set of methods, collectively called *properties*, in addition to the roles involved. Association classes are used to capture the properties of associations among objects with specific roles. The number of roles participating in an association is called the *arity* of the association. Association classes among the same set of classes may form a hierarchy.

Fig. 2a shows the student-employee example with associations for Taking and Offering courses. Note that Student and Employee are both roles of Person and roles in associations Taking and Offering. It also shows that a ProjectManager is a role of an Employee. Fig. 2b shows an example of medical image feature relationships for the tumor in a patient's brain, where a microlesion evolves to macrolesion and its spatial relationship with the lateral ventricle changes from disjoint to invading.

(a) Roles and associations of students and employees.　(b) A growing tumor invades the lateral ventricle in a brain.

□ Entity　⌐¬ Role　◇ Association　—— Association Link　- - -> RoleOf　Multiplicity (min, max) *: many
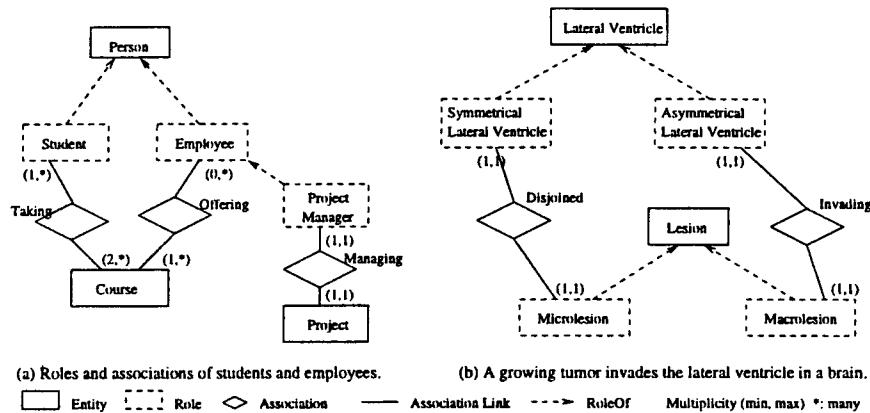
Fig. 2. Examples modeled by entity-role-association constructs.

The basic integrity constraints related to associations are:

1. Referential integrity: the role objects referenced in an association must exist and have the right types. This is a constraint on individual instances.
2. Multiplicity: the number of association instances of the same class in which each role can participate. The multiplicity of each role is specified with a (*min*, *max*) pair (as shown in Fig. 2), meaning that a role has to participate in the associations at least *min* times and at most *max* times. The multiplicity is a constraint on collections of instances.

Based on the properties, associations can be divided into different kinds: (1) Simple associations: A simple association is a simple link that does not have any particular properties of its own. An example of a simple association is *PartOf*

relationship. (2) Complex associations: A complex association is one that has its own properties. Instances of associations are used to capture the interrelations among objects. (3) Polymorphic associations: Polymorphic associations are the same kind associations, but for different role classes they have different properties. These associations classes constitute a class hierarchy. For an association instance and a given message to it, the behavior depends on the types of roles involved in the association. For instance, a borrower borrowing a loan item from a library is modeled by `Borrowing` association class, and `loan_period()` is a method of the association. `Loan_period()` depends on the borrower classes and the loan item classes.

The entity-role-association scheme provides flexible basic model constructs. Other high-level objects can be built on top of these basic constructs, e.g. by grouping related objects. As a result, it can be used to specify dynamic and evolving OO systems.

3 Association Classes and Their Features

We first propose a scheme for the association class definition, then discuss the object-oriented characteristics for the association classes. To focus on the association construct and its properties, we ignore the RoleOf relationship on the role class side in this section. We use a simple library model as shown in Fig. 3 to illustrate association features.
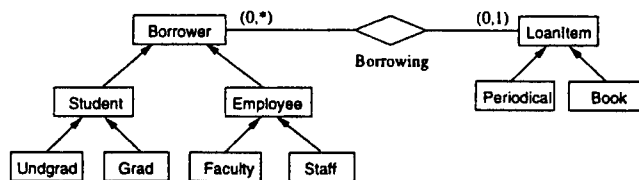
Fig. 3. A simple library model with association features.

An association class can be defined with its role classes, instance variables, and methods. For example, Fig. 4 shows for the `Borrowing` association class between `Borrower` and `LoanItem` for the library example in Fig. 3. The role classes in the order they appear in the association class definition constitute the *tuple of role classes* for the association.

All the association classes in a hierarchy for the "same" type of associations have the same name. For instance, if we want to capture the specialties of the association in different cases. For example, the "same" methods `loan_period` for `Borrowing[Borrower, Periodical]` and `Borrowing[Faculty, Book]` are different, and they would be defined as association classes as follows:

`association Borrowing[Borrower, LoanItem]{...};`

```
// association class name and the tuple of role classes:
association Borrowing[Borrower, LoanItem]
{   // normal attributes
    Date    loanDate;
    Date    dueDate;
    Date    returnDate;
    // methods
    virtual int loan_period() { ... }
};
```

Fig. 4. An association class definition for the simple library model example.

```
association Borrowing[Borrower, Periodical]:
            Borrowing[Borrower, LoanItem]{...};
association Borrowing[Faculty, Book]:
            Borrowing[Borrower, LoanItem]{...};
```

We now discuss the characteristics of object-orientation for the association classes. The major differences between association classes and regular classes are on the classification and polymorphism.

The constructors for association instances require a tuple of role instances that corresponds to the tuple of role classes declared in the association class. It is feasible that the constructors of the association classes are polymorphic, based on the role classes. With polymorphic constructors, it is unnecessary to do typecase testing on roles when constructing association objects. It puts more restrictive classification limit on the association instances.

We shall present two other major features – inheritance and polymorphism in the following sections.

3.1   Inheritance of Association Classes

One usage of inheritance is to specialize a class while allowing the subclass to inherit common properties from its superclass. Association class inheritance has its own features.

Implicit Inheritance. An association class defined for some role classes in class hierarchies is applicable to tuples of their corresponding subclasses.

For the library example (Fig. 3), because [Undgrad, Book] ≺ [Borrower, LoanItem], the association class Borrowing[Borrower, LoanItem] defined in Fig. 4 is applicable to tuple [Undgrad, Book]. Similarly, the same holds for tuple [Grad, Periodical].

Explicit Inheritance. The explicit inheritance of association classes is mainly used to override the implicitly inherited association class properties, therefore

the following special rules apply to the explicit association class inheritance: (1) The subclass must have the same name as the superclass. (2) The subclass has the same arity as the superclass. (3) The tuple of role classes in the subclass are more specific than that of the superclass.

The following example shows how a subclass can be defined.

```
association Borrowing[Undgrad, Book] :   // subclass
            Borrowing[Student, LoanItem] // superclass
{   ...
    virtual int loan_period() { ... }   // overriding
};
```

3.2 Multiple Polymorphism of Messages on Associations

Given a message on an association instance, there are the following two message dispatching cases:

1. If the method is explicitly defined on the association class of the instance, then the defined method is invoked.
2. If the method is not explicitly defined on the corresponding class of the instance, then the implicitly inherited method will be invoked.

The method invoked depends on the role class tuple of the association instance, thus is *multiply polymorphic*. To determine a unique method to invoke for a message, the explicitly defined association methods must satisfy the unambiguity requirement. For the library example, if we have loan_period() explicitly defined on Borrowing[Borrower, Book] and on Borrowing[Undgrad, LoanItem], to avoid ambiguity, loan_period() must be explicitly defined on Borrowing[Undgrad, Book]. An algorithm can help check if this requirement is met.

For any polymorphic methods that satisfy the unambiguity requirement, we can determine a unique and most specific method to invoke for a message.

4 The Relationship Between Associations and Roles

When implementing associations, depending on the requirements and constraints, there are two alternatives: intrinsic associations and extrinsic associations. Intrinsic associations provide a direct navigation path from role instances to the association instances, while extrinsic associations allow new associations defined on existing objects. Both alternatives can be realized as extensions atop "traditional" OO systems.

4.1 Intrinsic and Extrinsic Associations

In the intrinsic associations (Fig. 5a), roles are tightly coupled with the associations they participate in. An instance variable within each of the role classes

is used for each association class, either single-valued or set-valued, depending on the multiplicity of the role in the association. It is similar to the traditional instance variable approach except that the instance variable now refers to the association objects.

(a) Intrinsic association   (b) Extrinsic associaiton
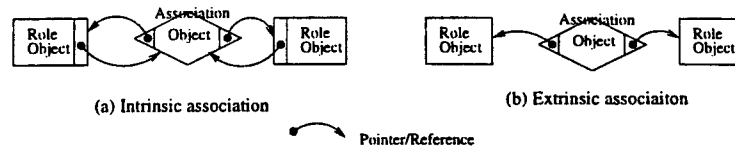
Fig. 5. Intrinsic and extrinsic binary association instances with roles.

If the role classes have already been defined and we want to add more associations but do not want to make changes to the role classes, or do not want to add additional variables and methods to role classes for sparse associations, we can use the extrinsic association (Fig. 5b). In such associations, no variables are used within role classes for associations, instead, a collection of all association instances is maintained, and the relationships from role objects to association objects is recovered by explicit join between role objects and the association instance collection. We call a collection of all the instances of the same association class an *association extent*.

For example, we can maintain the collection for the Borrowing associations, which can be defined or generated like this:

```
Multiplicity multi[] = {MANY, MANY};
extent<Borrowing> borrowing(multi);
                // for a many-to-many association Borrowing
```

An association extent is a special collection class with integrity enforcement. The association extent also needs to support associative search, join with roles, and probably aggregate functions.

The advantages for the extrinsic association are flexibility in adding new associations to existing classes, and centralized integrity maintenance for each association class. Extrinsic associations also reduce code dependencies among modules using the same roles. Polymorphism can be utilized without adding methods into existing classes.

4.2 Interdependencies Between Roles and Associations

The referential integrity requires that the roles referenced in associations exist. Therefore when an association instance is to be created, the roles either exist or should be created at the same time. When a role is deleted, the deletion is cascaded to all the associations that the role is involved. An application may require that the deletion of a role be prohibited if there are still existing associations involving the role.

On the other hand, a role may exist without participating in any associations, that is, in the multiplicity of the role $min = 0$. For example, the borrower role for a student belongs to this case, because a student is a borrower even he is not currently borrowing any books. A role may be required to be involved in associations for the role to exist, i.e. $min > 0$ for the role. A project manager role in Fig. 2 belongs to this case, since a project manager is an employee who manages a project.

Since there are references to roles in associations, the navigation from an association to roles is always direct. However, navigation from a role to the association may not always be possible. Depending on the implementation scheme, the navigation from a role to associations can be direct (intrinsic associations) or indirect (extrinsic associations with extents).

5   The Relationship Between Players and Roles

A player of a role can be an entity or another role, which is ultimately played by an entity. Depending on the situation and functionality requirements, a role may be incorporated into an entity as a base role, and an entity may or may not be aware of some or all of the roles it is currently playing. An entity may be only shared by the extrinsic roles for redundancy elimination.

A base role is a role played by an entity for its entire lifetime in the system. Thus, if the entity is not playing the base roles, it would not be included into the system. Therefore there is no need to separate the base roles from the entity. For example, Employee is usually considered as a base role and is not separated from Person in a company personnel system.

Often it is unnecessary for the player to keep track of all the roles it is playing. Thus, there are links only from the roles to the player. The roles cannot be accessed directly from their player. We call these *extrinsic roles* to the player (Fig. 6a). The typical supporting scheme for extrinsic roles is the inheritance by delegation [Taiv96]. That is, messages that are not part of direct interface of the role are forwarded (delegated) to the player. The user of the role object treats the role instance the same way as the role class inherits the player class in subclassing. For instance, a student is a member of the student body, but he is also a member of a club. The club member role can be modeled by an extrinsic role.

When a player needs to know the specific roles that it is playing, the roles are *intrinsic*, and the behavior of the player depends on these roles. The roles may be only referenced by the player, i.e. unidirectional from the player to the roles. In this case, roles are just part of states of the player (Fig. 6b). When the relationship between the player and roles are bidirectional (Fig. 6c), in addition to being accessed from inside the player object, the intrinsic roles can also be referenced by other objects. For example, the properties of an employee and his project manager role (Fig 2a) are dependent on each other, therefore, a bidirectional intrinsic role for project manager should be used (as in Fig 6c).

(a) Extrinsic Role   (b) Unidirectional Intrinsic Role   (c) Bi-directional Intrinsic Role

●——► Pointer/Reference
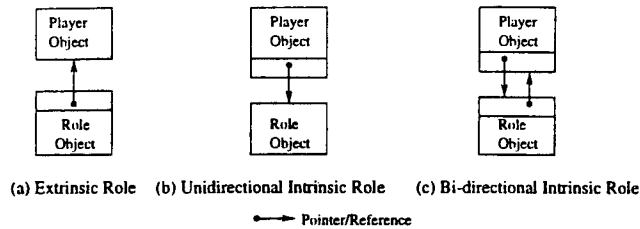
Fig. 6. Extrinsic and intrinsic roles.

The relationships between players and roles are either one-to-one or one-to-many, with roles dependent of players, i.e. a role cannot exist without a player. Since role classes cannot be instantiated without a player, players always exist first. An entity can play multiple roles at the same time, and play different sets of roles at different times.

The dependencies among players, roles, and associations have to be defined precisely in the system model to achieve system integrity.

6   An Application Example

Let us use our experience in implementation of our CoBase [Chu96] project as an example to illustrate the usefulness of the association class and role class. CoBase is a cooperative database interface which provides query relaxation (approximate answers), associative querying (provides relevant answers that the user does not explicitly ask for), and explanation that describes the relaxation process and its reasoning. A portion of its classes and associations are shown in Fig. 7, where RelaxEngine is for Relaxation Engine, AssocEngine is for Association Engine, and ExplanEngine is for Explanation Engine. These engines all work on a uniform internal query representation (QueryRep). Part of QueryRep component classes are shown in the left-hand side of Fig. 7.

There is a large class hierarchy for representing operands in SQL query conditions with our cooperative SQL (CoSQL) [CID96] extension. Engine modules of the system manipulate on a query representation (QueryRep) to achieve their functionalities. The CoBase system is developed incrementally. In traditional object-oriented way, many functions would have to be added gradually into the operand class hierarchy. It would cause endless recompilations of all the code and a fat interface that is the union of relatively independent functions for different modules. If we extend functions outside of the QueryRep, we have to use a long list of typecase tests to discriminate the types of the operands. Polymorphism is important in eliminating these long lists of cases.

To reduce the dependencies among RelaxEngine, AssocEngine, and ExplanEngine, we used the following mechanisms:

1. Extrinsic role classes for QueryCond classes are used for roles associated with RelaxEngine and AssocEngine to keep their states. Extrinsic roles make

their usage of `QueryCond` in `RelaxEngine` and `AssocEngine` independent of each other.

2. Extrinsic roles are used for the conditions that need to be explained in `ExplanEngine`. As a result, there are only dependencies of `ExplanEngine` on `RelaxEngine` and `AssocEngine`, but not the other direction.

3. Intrinsic associations are used for the associations on the `RelaxEngine`, `AssocEngine`, and `ExplanEngine`, so that the manipulation of these associations is direct from these respective engines.
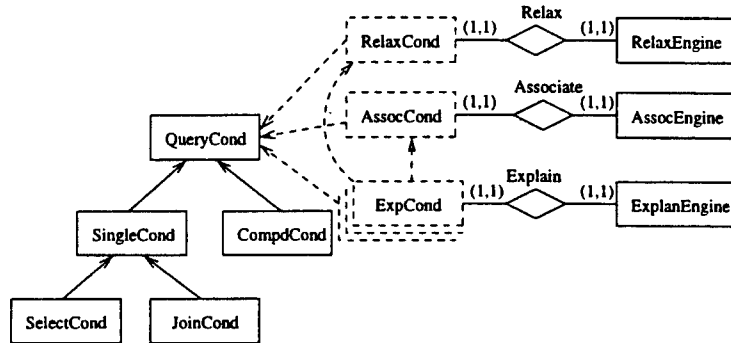
Fig. 7. Illustration of associations and roles in part of the CoBase system.

Our experience has revealed the association and role constructs not only increase expressive power, but more importantly, greatly simplify system development and maintenance.

7 Related Work

The association construct proposed in this paper is related to semantic data modeling, object modeling technique(OMT), multi-methods in functional OO languages, and traditional relational database integrity constraints.

The importance of supporting explicit relationships in object-oriented systems has long been recognized [Ditt90, Rumb87, Tan95]. There have been also several attempts in supporting relationships in object-oriented databases [AGO91, Brat91, Catt96, DG91, NQZ91]. In [Rumb87], Rumbaugh proposed relation as a construct to support associations. A relation is a collection of associations, but associations are not objects and cannot have methods. The relationship support proposed in [Brat91] is similar on this aspect. In [AGO91], an association is a set of distinct bindings, each of which is like a value tuple. Their association is more like our association extent, or an extension to relations for relationships in relational databases. The major difference is that our association also emphasizes the properties of individual association instances. ODMG-93 [Catt96] has a `relationship` prefix for instance variables that are for associations to maintain

integrity constraints, but no special construct for associations. An association concept in OMT is proposed by Rumbaugh et al.[Rumb91] and used in object relationship modeling and translation into regular instance variables. These latter two proposals suffer some of the problems as pointed out in the introduction of this paper.

Our work is similar to those in semantic data modeling[Chen76, HK87, PM88, TYF86], which focuses on structural aspects of representing semantics in different constructs. However, we introduced behavior modeling with object-oriented constructs.

Multi-methods in OO languages[BK86, Cham92, MHH91] provide multiple polymorphism and is a well-studied topic for multiple polymorphism and dispatching. But they are not as well organized and have been criticized as not being object-oriented, since they are not tied with any objects. Association class construct makes multiple polymorphism a natural property of association instances, an extension to the current object-oriented model.

Object-role modeling [Bron95, SD96] allows bottom-up analysis, based on local relationship properties. Then the roles can be integrated into classes. Object with roles extension to object-oriented systems [GSR96, RS91, Pern90, Alba93, WCL97] aims at solving the object evolution problem, thus provides more flexibility to the class-based systems. We are unifying them under the entity-role-association scheme, combining the benefits from both.

8   Conclusions

We have proposed entity-role-association as basic constructs for supporting flexible applications and studied the properties and relationships of these constructs.

The ISA relationships among classes are subdivided into ISA and RoleOf relationships. ISA is used for static classification and RoleOf is for dynamic role-playing and object evolution.

A new association class construct is proposed. Its object-oriented features are presented. ISA relationship in association class hierarchy has its special features. Different implementation scheme of associations meet different needs. The intrinsic association allows easy referential integrity maintenance, while the extrinsic association allows easy extension for existing classes and objects. The extrinsic association also allows polymorphism without adding methods into role classes, which can improve the code dependencies among modules.

The role concepts from *object with roles* and *associations* are unified based on the observation that entities play roles in association with other entities. The roles are classified into extrinsic roles and intrinsic roles. Extrinsic roles provide easy extension to existing objects, while intrinsic roles allow players to adapt their behaviors based on the roles they are playing. The different role classification allow developers to choose suitable roles in their implementation.

The entity-role-association scheme supports bottom-up analysis and easy integration of submodels. The proposed constructs not only provide clear semantics, clear interdependencies, and guidance for the implementation with current

OO techniques, also can be supported as language constructs. Our experience shows that our proposed model provides useful software structure, and also results in easy software maintenance.

Acknowledgments

References

[AGO91]   Albano, A., Ghelli, G., and Orsini, R., A relationship mechanism for strongly typed object-oriented database programming language, in *Proceedings of the 17th VLDB conference*, Barcelona, Sept. 1991, pp.565-575.

[Alba93]   Albano, A. et al. An object data model with roles, *Proc. 19th VLDB Conf.*, Dublin, Ireland, 1993, pp. 39-51.

[BK86]   Bobrew, D. G., and Kahn, K. et al. CommonLoops: Merging lisp and object-oriented programming, In *Proceedings of ACM OOPSLA '86*, pp. 17-29.

[Booc94]   Booch, G., *Object-oriented analysis and design with applications*, Benjamn/Cummings, Redwood City, CA, 1994.

[Brat91]   Bratsberg, S. E., FOOD: Supporting explicit relations in a fully object-oriented database, in *Object-oriented databases: Analysis, design & construction (DS-4)*, Proceedings of the IFIP TC2/WG 2.6 Working Conference, Windermere, UK, July 1990, pp. 123-140.

[Bron95]   Bronts, G.H.W.M. et al., A unifying object role modelling theory, *Information Systems*, Vol. 20, No. 3, 1995, pp. 213-235.

[Catt96]   Cattell, R. G. G. (Ed.), *The object database standard: ODMG-93 Release 1.2*, Morgan Kauffmann Publishers, Inc. San Francisco, California, 1996.

[Cham92]   Chambers, C., Object-oriented multi-methods in Cecil, in *Proceedings of ECOOP '92*, LNCS 615, pp. 33-56.

[Chen76]   Chen, P. P., The entity-relationship model: toward a unified view of data, *ACM TODS* Vol.1, No.1, March, 1976, pp.9-36.

[Chu96]   Chu, W. W., Yang, H., Chiang, K., Minock, M. Chow, G., Larson, C., CoBase: A Scalable and Extensible Cooperative Information System, *Journal of Intelligent Information Systems*, 1996.

[CID96]   CoBase Internal Document, CoSQL Specification Report, CoBase Research Lab., Computer Science Department, UCLA, 1996.

[DG91]   Diaz, O. and Gray, P. M. D., Semantic-rich user-defined relationship as a main constructor in object-oriented databases, in *Object-oriented databases: Analysis, design & construction (DS-4)*, Proceedings of the IFIP TC2/WG 2.6 Working Conference, Windermere, UK, July 1990, pp. 207-224.

[Ditt90]   Dittrich, K. R., Object-oriented database systems: the next miles of the marathon, *Information Systems*, Vol. 15, No. 1, pp. 161-167, 1990.

[GHJV95]   Gamma, E., Helm, R., Johnson, R. and Vlissides, J., *Design patterns: elements of reusable object-oriented software*, Addison-Wesley, Reading, MA, 1995, pp.331-344.

[GSR96]   Gottlob, G., Schrefl, M. and Röck, B., Extending object-oriented systems with roles, *ACM TOIS*, Vol. 14, No. 3, July 1996, pp.268-296.

[HK87]   Hull, R. and King, R., Semantic database modeling: survey, applications, and research issues, *ACM Computing Surveys*, Vol. 19, No.3, Sept. 1987, pp. 201-260.

[Ing86]   Ingalls, D. H. H., A simple technique for handling multiple polymorphism, in *OOPSLA '86 Proceedings*, Sept. 1986, pp. 347-349.

[MHH91]   Mugridge, W. B., Hammer, J. and Hosking, J. G., Multi-Methods in a statically-typed programming language, in *Proceedings of ECOOP '91*, pp.307-324.

[NQZ91]   Nassif, R., Qiu, Y., and Zhu, J., Extending the object-oriented paradigm to support relationships and constraints, in *Object-oriented databases: Analysis, design & construction (DS-4)*, Proceedings of the IFIP TC2/WG 2.6 Working Conference, Windermere, UK, July 1990, pp. 305-330.

[Pern90] Pernici, B., Objects with Roles, *Proc. ACM Conf. on Office Information Systems*, 1990, pp.205-215.

[PM88]   Peckham, J. and Maryanski, F., Semantic data models, *ACM Computing Surveys*, Vol.20, No. 3, Sept. 1988, pp. 153-189.

[Rat97]   Rational Software Corporation, UML semantics v1.0, Jan. 1997.

[RS91]   Richardson, J. and Schwarz, P., Aspects: Extending objects to support multiple, independent roles, *Proc. ACM SIGMOD '91 Conf.*, May 1991, pp. 198-307.

[Rumb87] Rumbaugh, J., Relations as semantic constructs in an object-oriented language, in *OOPSLA '87 Proceedings*, Oct. 1987, pp. 466-481.

[Rumb91] Rumbaugh, J. et al., *Object-oriented modeling and design*, Prentice-hall, Englewood Cliffs, New Jersy, 1991.

[SD96]   Snoeck, M. and Dedene, G., Generalization/specialization and role in object oriented conceptual modeling, *Data & Knowledge Engineering*, Vol. 19, 1996, pp. 171-195.

[Taiv96] Taivalsaari, A., On the Notion of Inheritance. *ACM Computing Surveys*, Sept. 1996, Vol. 28, No. 3, pp.438-479.

[Tan95]   Tanzer, C. , Remarks on object-oriented modeling of associations, *JOOP*, Vol. 7, No. 9, Feb. 1995, pp. 43-46.

[TYF86]   Teorey, T. J., Yang, D. and Fry J. P., A logical design methodology for relational databases using the extended entity-relationship model, *ACM Computing Surveys*, Vol. 18, No.2, June 1986, pp.197-222.

[WCL97]   Wong, R. K., Chau, H. L., and Lochovsky, F. H., A data model and semantics of objects withdynamic roles. In *13th IEEE International Conference on Data Engineering*, April, 1997.

Explanation Over Inference Hierarchies in Active Mediation Applications

Michael Minock and Wesley Chu
Computer Science Department
University of California, Los Angeles, CA 90024
mjm@mcc.com wwc@cs.ucla.edu

Abstract

This paper gives algorithms to compute the explanation of instance membership in classes over an Inference Hierarchy. $O(c)$ time algorithms are given to compute the positive, negative, and conditional explanations over an Inference Hierarchy limited to conjunctive parents, where c is the total number of conditions in the Inference Hierarchy[1]. Algorithms requiring $O(c^2 k)$ time and space are given for computing the k-minimal positive, negative and conditional explanations over disjunctive hierarchies with independent conditions. Worst-case exponential time is required for computing negative and negative conditional explanations over disjunctive hierarchies with dependent conditions. All types of explanation are worst-case exponential time for the k-minimal explanations over hierarchies using negation in parent relationships.

An application for these algorithms is provided. The application is motivated by the DARPA ALP storyboard and provides active mediation of airport facility status in a combat situation. These techniques may be transferred to financial, administrative, and environmental domains as well.

1 Introduction.

Active mediation systems monitor a dynamic pool of information and trigger actions as pre-defined rule conditions match[4]. As the complexity of the monitored information and the number of valid triggering conditions grows, it is necessary to be able to explain either *why* or *why not* active mediation

---

[1]The number of conditions corresponds, roughly, to the number of classes in the Inference Hierarchy.

rules match. Though there have been attempts to endow intelligent systems with explanatory capabilities[11][9][10], little has been done to guarantee that explanations are complete and correct. This paper defines a mechanism for providing complete and correct explanation of positive, negative, and conditional instance membership in class nodes of an Inference Hierarchy, and then applies this mechanism toward the problem of active mediation of airport facility status in a combat situation.

Section 2 of this paper gives the definition of the Inference Hierarchy limited to conjunctive parents. Section 3 presents algorithms for computing positive, negative, and conditional explanation of instance membership in classes over Inference Hierarchies limited to conjunctive parents. Section 4 extends the Inference Hierarchy to include disjunctive parents and presents the necessary extensions to provide positive, negative, and conditional explanation of instance membership. Section 5 further extends the Inference Hierarchy to include negation in parent relationships. Section 6 presents a DARPA ALP active mediation example. Section 7 compares this work to previous work.

2   Definition of the Inference Hierarchy

The Inference Hierarchy is a network of class nodes, where edges represent parent relationships. A class node with multiple parents requires that an instance be a member of all its parents[2] to be a member of the class. In addition, each class node has a conjunction of conditions which an instance must meet to be a member of the class. Finally the Inference Hierarchy is a connected directed acyclic graph and requires a unique source node representing the universal class.

2.1   Formal Definition of the Inference Hierarchy

Definition 1 *(Class Node) A class node is N where $N =< id, (P_1, ..., P_r), (C_1, ..., C_t) >$. Where id is the identifier of the node, P is a parent node identifier, and C is a condition predicate that instances of N must satisfy.*

Definition 2 *(Inference Hierarchy) An Inference Hierarchy H is a topologically s~rted vector of class nodes $[N_1, ..., N_n]$ where each N is a class node definition.*

---

[2]This conjunctive requirement will be generalized to include disjunction and negation of parent membership.

The notation for the i-th node in H is $N_i = < id_i, (P_{i,1}, ..., P_{i,r_i}), (C_{i,1}, ..., C_{i,t_i}) >$. The following conditions apply to H:

(1) $id_i = id_j \Longrightarrow i = j$

(2) $P_{i,k} = id_j$ where $1 \leq k \leq r_i \Longrightarrow j < i$.

(3) $t_1 = 0$, $r_1 = 0$ and $r_i \geq 1$ for $i \geq 1$

Property (1) guarantees that all class identifiers are unique. Property (2) guarantees that the hierarchy is acyclic. Property (3) guarantees that the hierarchy has a unique 'universal' source node and the hierarchy is fully connected.

2.2 Classification under a Conjunctive Inference Hierarchy

An instance τ may be a member of any subset of class nodes. Presuming that all conditions of all class nodes may be evaluated to *true* or *false*[3] on τ, there is an $O(n)$ algorithm to compute class membership over an n-class Inference Hierarchy.

The classification of an instance proceeds by iterating over all classes in the order of their definition. Because H is topologically sorted, the membership of all parent classes will have been established, so the parent membership expression may be evaluated to determine if the instance might be a member of the class. If so, the instance is subjected to the evaluation of the conditions of the class. If the instance meets these conditions then the instance is marked as a member of the class.

Definition 3 *(Instance Classification) The following algorithm classifies the instance τ over the Inference Hierarchy $H = [N_1, ..., N_n]$:*

```
Classify(τ, [N₁, ..., Nₙ]) {
    for (i = 1 to n) {
        if (τ ∈ Pᵢ,₁ ∧ ... ∧ τ ∈ Pᵢ,ᵣᵢ ∧ Cᵢ,₁(τ) ∧ ... ∧ Cᵢ,ₜᵢ(τ))
            then τ ∈ idᵢ
            else τ ∉ idᵢ }
}
```

---

[3]Note that this assumes the closed world assumption[8].

3 Explanation of Classification

3.1 Positive Membership

The Question to be answered is "why is instance τ a member of class X?", signified here as $why(\tau \in X)$.

Definition 4 *(explanation of positive membership) The set C is an explanation of $\tau \in X$ iff for all $c \in C$, $c(\tau)$ and for arbitrary instance τ', for all $c \in C$, $c(\tau') \Longrightarrow \tau' \in X$.*

Theorem 1 *For inference hierarchies limited to conjunction,*

$$why(\tau \in id_i) = \{C_{i,1}, ..., C_{i,t_i}\} \cup why(\tau \in P_{i,1}) \cup ... \cup why(\tau \in P_{i,r_i}) \quad (1)$$

is the unique, minimal explanation of $\tau \in id_i$.

Proof:

By definition $why(\tau \in id_i) = C_{ancestors(id_i)}$ where $C_{ancestors(id_i)}$ are all the conditions of all the class nodes above id_i including $\{C_{i,1}, ..., C_{i,t_i}\}$. Assume that an explanation for $(\tau \in id_i)$ included c where c is a condition on a class node in H and $c \notin C_{ancestors(id_i)}$. By definition of the classification algorithm, this c is irrelevant to the classification of $\tau' \in id_i$. Hence this c need not be included in the explanation of $\tau \in id_i$. Thus an explanation requires only conditions in $C_{ancestors(id_i)}$.

Assume that an explanation does not contain c where $c \in C_{ancestors(id_i)}$. This explanation allows for $\neg c$, which based on the classification algorithm, gives $\tau' \notin id_i$. Thus an explanation includes every condition in $C_{ancestors(id_i)}$. Putting it together, the unique and minimal[4] explanation of $\tau \in id_i$ is $C_{ancestors(id_i)}$. •

3.2 Negative Membership

The question "why is instance τ not a member of class X" signified $why(\tau \notin X)$ may also be answered[5].

---

[4]This minimality is absolute for Inference Hierarchies with independent conditions. Otherwise, a simple constraint reduction mechanism may be applied to this set to derive the absolute minimal set of conditions. For example *salary > 10k*, *salary > 20k* would reduce to *salary > 20k*.

[5]The integrity of this mechanism depends on the quality of the Inference Hierarchy. All classes in the Inference Hierarchy must be 'satisfiable', that is for each class X there must

Definition 5 *(explanation of negative membership)*
  The set C is the explanation of $\tau \notin X$ iff for all $c \in C$, $\neg c(\tau') \Longrightarrow \tau' \in X$.

That is, a negative explanation is a set of condition predicates that if reversed on τ, would imply $\tau \in X$.

Theorem 2

$$why(\tau \notin id_i) = \{C'_1, ..., C'_j\} \cup why(\tau \notin P'_1) \cup ... \cup why(\tau \notin P'_k) \qquad (2)$$

is the unique, minimal[6] explanation for $\tau \notin id_i$ where $C'_1(\tau)...C'_j(\tau)$ are <u>all</u> the conditions of id_i that evaluate to false on τ and $P'_1...P'_k$ are <u>all</u> the parents of id_i for which τ is not a member.

Proof:

By definition $why(\tau \notin id_i)$ are exactly those conditions $c \in C_{ancestors(id_i)}$ for which $\neg c(\tau)$. If all the conditions of $why(\tau \notin id_i)$ were reversed then all $c \in C_{ancestors(id_i)}$ would evaluate to $true$[7]. Based on the classification algorithm, this would imply $\tau \in id_i$. Hence $why(\tau \notin id_i)$ is an explanation of $\tau \notin id_i$. It is minimal because if any condition in $why(\tau \notin id_i)$ is not reversed then $\tau \notin id_i$. •

3.3 Conditional Explanation

The explanation $why(\tau \in X)$ will include all the conditions that establish τ as a member of X. In a large Inference Hierarchy such explanations will be too long. Users are not usually interested in this level of detail. The user has in mind a more precise question. That is $why(\tau \in X | \tau \in Y)$ or

---

be an assignment of conditions that classify an instance τ' into X. This is a reasonable requirement for an Inference Hierarchy. If no instance could possibly be a member of a class, why represent the class? (Even if the class represents an integrity constraint, representing the integrity constraints acknowledges the possibility, that because of improper data, an instance may be inferred to be a member of the 'impossible' class). The reason why the hierarchy must be satisfiable is because there must be some assignment of conditions on an arbitrary instance τ that for an arbitrary class X classifies $\tau \in X$. Otherwise explanation of negative class membership will search for such satisfiability and fail after an exponential number of condition assignment combinations.

[6]Minimal in the sense of reporting the fewest number of conditions that would require reversal for $\tau \in id_i$ to be true.

[7]Based on the stipulation that classes in the hierarchy must be satisfiable, the interpretation where the non-explanatory conditions on τ remain *true* is consistent.

$why(\tau \notin X | \tau \in Y)$. These conditional type questions enable the user to request more concise explanations[8].

The interpretation here is that the user fully understands, that is has all explanations for, $why(\tau \in Y)$[9]. The user is interested in what additional information "explains" $\tau \in X$ or $\tau \notin X$.

3.3.1 Algorithms to Compute Conditional Explanation

These algorithms give an efficient method for computing conditional explanations by extracting a portion of the Inference Hierarchy, and computing standard positive or negative explanation over that portion.

The algorithm to compute $why(\tau \in X | \tau \in Y)$:

(1) if $\tau \notin Y$ then $misconception(\tau \in Y)$.

(2) if $\tau \notin X$ then $misconception(\tau \in X)$.

(3) $Anchor = lub(X, Y)$.[10]

(4) Derive $X_{support}$: the subgraph of the nodes containing all paths from X to $Anchor$ through the parent relationship.

(5) Derive $Y_{support}$: the subgraph of the nodes containing all paths from Y to $Anchor$ through the parent relationship.

(6) if $X_{support} \subset Y_{support}$ then
$misconception(\tau \in Y \Rightarrow \tau \in X)$ else
$why(\tau \in X \, over\{X_{support} - Y_{support}\})$ [11]

The algorithm to compute $why(\tau \notin X | \tau \in Y)$:

(1) if $\tau \notin Y$ then $misconception(\tau \in Y)$.

(2) if $\tau \in X$ then $misconception(\tau \notin X)$.

---

[8]The basic question $why(\tau \in X) \equiv why(\tau \in X | \tau \in id_1)$ and $why(\tau \notin X) \equiv why(\tau \notin X | \tau \in id_1)$.

[9]This interpretation applies when the Inference Hierarchy is extended to include disjunction and negation.

[10]$lub(X, Y)$ is the least upper bound of class X and class Y. This is defined as the common ancestor of X and Y through which all paths from X to id_1 and all paths from Y to id_1 pass where there is no descendant with the same property.

[11]Where $over$ specifies the sub-hierarchy over which to compute standard explanation. $X_{support} - Y_{support}$ obtains the sub-hierarchy rooted at $Anchor$ containing all reachable nodes of $X_{support}$ given that no nodes of $Y_{support}$ may be traversed.

(3) $Anchor = lub(X, Y)$.

(4) Derive X_{refute}: the subgraph of the nodes containing all paths from X to *Anchor* through the parent relationship.

(5) Derive $Y_{support}$: the subgraph of the nodes containing all paths from Y to *Anchor* through the parent relationship.

(6) if $X_{refute} \subset Y_{support}$ then $misconception(\tau \in Y \Rightarrow \tau \notin X)$ else $why(\tau \notin X over\{X_{refute} - Y_{support}\})$

4 Conjunction and Disjunction Inference Hierarchies

The extension to disjunction in addition to conjunction is the next[12] step in making the Inference Hierarchy more expressive. This is achieved by extending the formal definition of the conjunctive system to encompass disjunctive parent relationships.

Definition 6 *(Class Node) A class node is N where $N = < id, ((P_{1,1}, ..., P_{1,r_1}), ... , (P_{d,1}, ..., P_{d,r_d})), (C_1, ..., C_t) >. .*

Here the parent expression is in disjunctive normal form where the parent expression is a disjunction of d disjuncts of conjunctions of parents class identifiers. The notation for the i-th node in H is $N_i = < id_i, ((P_{i,1,1}, ..., P_{i,1,r_{i,1}}), ..., (P_{i,d_i,1}, ..., P_{i,d_i,r_{i,d_i}})), (C_{i,1}, ..., C_{i,t_i}) >$.

4.1 Classification

Definition 7 *(Instance Classification) The following algorithm classifies the instance τ over the Inference Hierarchy $H = [N_1, ..., N_n]$:*

```
Classify(τ, [N₁, ..., Nₙ]) {
      for (i = 1 to n) {
            if for some j, 1 ≤ j ≤ dᵢ, (τ ∈ Pᵢ,ⱼ,₁ ∧ ... ∧ τ ∈ Pᵢ,ⱼ,ᵣᵢ,ⱼ)
            ∧Cᵢ,₁(τ) ∧ ... ∧ Cᵢ,ₜᵢ(τ)
                  then τ ∈ idᵢ
                  else τ ∉ idᵢ }
      }
```

---

[12] Addition of negation to the conjunctive system would necessarily add disjunction. However adding disjunction does not add negation.

89

4.2 Explanation of Positive Membership

Theorem 3 *For inference hierarchies limited to conjunction and disjunction,*

$$why(\tau \in id_i) = \{C_{i,1}, ..., C_{i,t_i}\} \cup why(\tau \in P_{i,j,1}) \cup ... \cup why(\tau \in P_{i,j,r_{i,j}}) \quad (3)$$

where $\tau \in P_{i,j,1} \land ... \land \tau \in P_{i,j,r_{i,j}}$ for some j such that $1 \leq j \leq d_i$, is an explanation of $\tau \in id_i$.

Proof:

Assume for an arbitrary τ' that for all $c \in why(\tau \in id_i)$, $c(\tau')$ and $\tau' \notin id_i$. For $i = 1$ this is contradiction because by the classification algorithm $C_{1,1}, ..., C_{1,t_1} \Longrightarrow \tau' \in id_1$. For the induction hypothesis, assume that this is a contradiction for $i < k$. Now for $i = k$, $C_{i,1}(\tau'), ..., C_{i,t_i}(\tau')$. Hence for at least one of the parent identifiers id_k where $k < i$, for all $c \in why(\tau \in id_i)$, $c(\tau')$ and $\tau' \notin id_k$. This violates the induction hypotheses. Thus based on the definition of the classification algorithm for conjunction and disjunction hierarchies, $\tau' \in id_i$. Hence $why(\tau \in id_i)$ is an explanation of $\tau' \in id_i$. ●

4.3 Explanation of Negative Membership

Theorem 4 *For inference hierarchies limited to conjunction and disjunction, where the conditions of H are independent,*

$$why(\tau \notin id_i) = \{C'_1, ..., C'_j\} \cup why(\tau \notin P'_1) \cup ... \cup why(\tau \notin P'_k) \quad (4)$$

is an explanation for $\tau \notin id_i$ where $C'_1(\tau)...C'_j(\tau)$ are <u>all</u> the conditions in id_i that fail on τ and $P'_1...P'_k$ are <u>all</u> the parents in <u>one</u> of the failing disjuncts for which τ is not a member only if <u>all</u> disjuncts of the parent expression of id_i fail for τ[13].

Proof:

Assume that all the conditions $c \in why(\tau \notin id_i)$ were reversed for τ and all conditions in $C_{ancestors(id_i)} - why(\tau \notin id_i)$ maintained their values. For $i = 1$ based on the definition of the classification algorithm for conjunction and disjunction hierarchies, $why(\tau \notin id_1)$ is an explanation of $\tau \notin id_1$. For

---

[13]If not all disjuncts of the parent expression fail for τ, then simply $why(\tau \notin id_i) = \{C'_1, ..., C'_j\}$

the induction hypothesis, assume that $why(\tau \notin id_k)$ is an explanation of $\tau \notin id_k$ for $k < i$. Now for i, $why(\tau \notin id_i)$ maintains that the conditions $\{C'_1, ..., C'_j\}$ must be reversed for $\tau \in id_i$. In the case that at least on parent conjunct is true, $\{C'_1, ..., C'_j\}$ is the explanation of $\tau \notin id_i$. When all parent conjunctions are false, then based on the induction hypothesis why will return explanations of these failures, and based on the induction hypothesis such explanations will be correct. Thus, by induction, $why(\tau \notin id_i)$ is an explanation of $\tau \notin id_i$. •

It is no longer the case that there is a unique minimal explanation for $\tau \in id_i$. In the worst case at each step there are d_i choices to explain membership in id_i. Hence there are potentially an exponential number of individual explanations[14]. However using depth first search with branch and bound the k-minimal explanations may be found in reasonable time.

A better solution is to propagate the k-minimal explanations at the time of classification. With each step in the classification propagate the k-minimal explanations (positive or negative) up to that point and store temporarily with the corresponding class node. This will pass the k-minimal explanations to every node in the Inference Hierarchy in $O(c^2k)$ time and space.

Unfortunately for hierarchies with dependent conditions, the results of negative membership explanations must be verified when the Inference Hierarchy includes disjunction. This leads to a worst-case of exponential time to find the k-minimal negative explanations.

4.4 Conditional Explanation

The conditional explanation algorithm may be applied almost unchanged to hierarchies with disjunction. A new requirement is that nodes in $X_{support}$, $Y_{support}$ must have τ as an instance[15]. The intuition here is that each class node in $X_{support}$ or $Y_{support}$ should indeed "support" the classification of $\tau \in X$ or $\tau \in Y$. Support is guaranteed through positive membership of τ. Note that X_{refute} is unchanged.

Once the portion of the Inference Hierarchy is limited, then the propagation of the k-minimal explanations via the classification algorithm may be run over this portion of the graph to give explanation in $O(c^2k)$ time and space.

---

[14]Consider a purely disjunctive Inference Hierarchy where τ is an instance of all classes. The number of explanations is equal to the number of paths through the Inference Hierarchy to the class membership being explained.

[15]This is always true in the conjunctive case.

5 Conjunction, Disjunction, and Negation Inference Hierarchies

All that is required to add negation is to add a sign s to each parent identifier in the disjunctive normal form expressing membership requirements among parents. A "+" sign indicates that an instance must be a member of the class, while a "−" sign indicates that an instance must not be a member of the class.

The notation for the i-th node in H is $N_i =< id_i, ((s_{i,1,1} \cdot P_{i,1,1}, ..., s_{i,1,r_{i,1}} \cdot P_{i,1,r_{i,1}}), ..., (s_{i,d_i,1} \cdot P_{i,d_i,1}, ..., s_{i,d_i,r_{i,d_i}} \cdot P_{i,d_i,r_{i,d_i}})), (C_{i,1}, ..., C_{i,t_i}) >.$

5.1 Classification and Explanation

The classification algorithm is the same as the classification algorithm for disjunction so long as signs are included and it is recognized that $\tau \in -X \equiv \tau \notin X$ and of course $\tau \in +X \equiv \tau \in X$.

As long as the sign bits are included, the standard positive and negative explanation algorithm for disjunction suffices as well. A complication however is that the results of these explanations will consist of literals rather than just atoms. The algorithm must simply embed signs in the conditions that they yield.

The results of negative membership explanations must be verified when the Inference Hierarchy includes negation. Because a positive explanation may include the negative explanation of one of its parents, this leads to a worst-case of exponential time to find the k-minimal positive or negative explanations.

5.2 Conditional Explanation

The computation of $X_{support}$ and $Y_{support}$ is complicated. Let us restrict the analysis to $X_{support}$. Because $\tau \notin id_j$ may "support" classifying $\tau \in X$, the simple rule in the disjunctive case is insufficient. $X_{support}$ must include all those class nodes which participate in classifying $\tau \in X$ over the Inference Hierarchy rooted at $lub(X, Y)$. This set may be computed by conducting a depth first search upward from X and including only those class codes that participate along a supporting explanation path terminating with $lub(X, Y)$. The same argument may be applied to $Y_{support}$.

There is one more wrinkle. It concerns X_{refute}. It may be that for $\tau \in X$ that $\tau \notin Y$. That is there might be some condition that established Y that, if reversed, would support X. The set that supports Y that if reversed would

support X id referred to as $Conflict_{X,Y}$. Such nodes must be included in the standard negative explanation. That is $why(\tau \notin XOver\{(X_{refute} - Y_{support}) \cup Conflict_{X,Y}\})$

To compute $Conflict_{X,Y}$ it is necessary to note the assumed signs in $Y_{support}$. In the depth first search to build X_{refute} signs are pushed. If the signs do not match $Y_{support}$ expectations, then the node is included in $Conflict_{X,Y}$.

6   Active Mediation Example Scenario

A practical application of the Inference Hierarchy is motivated by the DARPA ALP(Advanced Logistics and Planning) program. The ALP program is interested in providing real-time notifications of logistics and battle field events.

In the scenario here, a set of airports are automatically monitored and if certain conditions are met, then a message is immediately sent to appropriate personnel to give attention to a potential problem. The problems relate to over or under utilization of airport resources, bad weather at airports, enemy threats to airports, maintenance breakdowns at airports, and critical risk situations at airports. The personnel to be notified are the logistics officers responsible for maintenance at airports, the field generals responsible for the airports in their sectors, and finally the theater general responsible for the complete employment.

6.1   Scenario Database Schema Definition

Airports are identified through the key **ANUM** and are positioned at a latitude and longitude. The following *static*[16] relation is populated with the monitored airports in the scenario:

AIRPORTS (ANUM, location_name, lat, lon)

Given this static relation of airports, reports are continuously released on the conditions at these airports and activity of enemy units. Reports are released for weather conditions, maintenance records, status (in terms of how many runways are available and how much fuel is available), and inventories at airports. These reports on airports are in the following database relations:

---

[16]There is only one static relation allowed per active mediation scenario. This static relation holds the information about the entities that messages are generated for – Airports in the scenario here.

```
AIRPORT_WEATHER_RPT( ANUM, wind_speed, temperature,
                     precipitation)
AIRPORT_MAINTENANCE_RPT( ANUM, num_disabled_aircraft)
AIRPORT_STATUS_RPT( ANUM, runways_available,
                     storage_space,fuel_supply)
AIRPORT_INVENTORY_RPT( ANUM, TYPE)
```

In addition to reports on airports, reports of enemy locations and force types are released. Enemy reports are represented by the relation:

```
ENEMY_ACTIVITY_RPT( FORCEID, force_type, lat, lon)
```

6.2 Domain Knowledge

6.2.1 Scenario Action Rules

Action rules are usually defined by commanders (or managers)– such authorities give broad definitions of who should be notified under high-level conditions. In the scenario here it is decided that logistics officers should be notified if there are maintenance crew problems at an airport, a field commander should be notified if an airport in their sector is over-utilized, under-utilized, or threatened, and the theater general should be notified if a critical situation develops. The rules expressing these actions are provided below.

```
NOTIFY-LOGISTICS-OFFICER if
   Airport is experiencing a MAINTENANCE_CREW_PROBLEM


NOTIFY-FIELD-COMMANDER if
   Airport is OVER_UTILIZED or UNDER_UTILIZED or THREATENED


NOTIFY-THEATER-GENERAL if
   Airport is in CRITICAL_SITUATION
```

6.2.2 High-Level Concepts

Given a set of action rules, domain experts are called upon to define the *high-level concepts* used in action rules. These definitions are in terms of other high-level concepts or are in terms of *conceptual values*. The high level concepts for our scenario are OVER_UTILIZED, UNDER_UTILIZED, THREATENED, MAINTENANCE_CREW_PROBLEM, and CRITICAL_SITUATION. The following rules define these concepts:

```
Airport is OVER_UTILIZED if FUEL_LOW or STORAGE_EXHAUSTED or
    BUSY_RUNWAYS

Airport is UNDER_UTILIZED if
    EXCESS_FUEL or EXCESS_STORAGE or EMPTY_RUNWAYS

Airport is THREATENED if
    EXTREME_WEATHER or BOMBER_THREAT or
    PANZER_THREAT or INFANTRY_THREAT

Airport has MAINTENANCE_CREW_PROBLEM if
    BAD_MAINTENANCE and UNDER_UTILIZED

Airport in CRITICAL_SITUATION if
    (THREATENED and OVER_UTILIZED) or
    (THREATENED and SENSITIVE_INVENTORY)
```

Because the system here allows high-level concepts to be defined in terms of other high-level concepts, a circularity in definition may occur. Such circularities are not allowed in the semantics here. Circularity in definitions is ruled out by requiring that rules are defined in order and all high-level concepts are defined before they are referenced in antecedents of rules. This is an example of imposing additional syntactic constraints on rule definitions to rule out circularity in the rule-base.

6.2.3   Conceptual Values

Finally the conceptual values of the domain must be defined. These are the only concepts which have conditions on the actual relational database. The definitions of the set of *conceptual values* required in this scenario are given:

```
BUSY_RUNWAYS if (AIRPORT_STATUS_RPT:runways_available < 2)

EMPTY_RUNWAYS if (AIRPORT_STATUS_RPT:runways_available > 8)

FUEL_LOW if (AIRPORT_STATUS_RPT:fuel_supply < 1,000 gal)

EXCESS_FUEL if (AIRPORT_STATUS_RPT:fuel_supply > 10,000 gal)

EXCESS_STORAGE if (AIRPORT_STATUS_RPT:storage_space > 10k m3)
```

```
STORAGE_EXHAUSTED if
    (AIRPORT_STATUS_RPT:storage_space < 100 m3)

BAD_MAINTENANCE if
    (AIRPORT_STATUS_RPT:num_disabled_aircraft > 10)

SENSITIVE_INVENTORY if
   (AIRPORT_INVENTORY_RPT:type in
   {'cipher machine','spy records','battle plans'})

EXTREME_WEATHER if
   (AIRPORT_WEATHER_RPT:wind_speed > 25km) or
   AIRPORT_WEATHER_RPT:precipitation > 1.4')

BOMBER_THREAT at ANUM if
   (ANUM NEAR_TO ENEMY_ACTIVITY_RPT:force_id and
   ENEMY_ACTIVITY_RPT:force_type = 'Bomber')

INFANTRY_THREAT at ANUM if
   (ANUM NEAR_TO ENEMY_ACTIVITY_RPT:force_id and
   ENEMY_ACTIVITY_RPT:force_type = 'Infantry')

PANZER_THREAT at ANUM if
   (NEAR_TO ENEMY_ACTIVITY_RPT:force_id and
   ENEMY_ACTIVITY_RPT:force_type = 'Panzer')
```

6.3 Action Inference Hierarchy

Given the action rule, high-level concept, and conceptual value definitions, it is necessary to build an *action inference hierarchy* that reflects this knowledge in an active mediation system. A strict requirement of such a system is that upon insertion, modification, or deletion of information in the database, if an action is licensed by the new information, that action must be executed immediately. The action Inference Hierarchy is used to compute which actions are to be performed when report tuples are INSERTED, MODIFIED, or DELETED.

6.3.1 Hierarchy Creation

The following algorithm constructs the action inference hierarchy given the database schema and the action, high-level concept, and conceptual value rule definitions provided by the domain expert:

(1) Create a node in the hierarchy for each action, high-level concept, and conceptual value rule. Define a database relation corresponding to that node. This relation, termed the *state relation*, is a relation over the key of the *static* relation.

(2) Add into each action node a DNF[17] parent expression reflecting the antecedents of the node's action rule. In each antecedent node place a *child* reference back to the given action node.

(3) Add into each high-level concept node a DNF parent expression reflecting the node's antecedents. In each antecedent node place a *child* reference back to the given high-level concept node.

(4) For each conceptual value node:

If the condition in the conceptual value rule references a database relation whose key is not the key of the *static* relation, then add this non-static key to the conceptual value node's state relation.

6.3.2 Trigger Insertion

(5) For each conceptual value node:

Place an ECA trigger into the database relation upon which the conceptual rule's conditions are defined. The *event* is INSERT or UPDATE. The *condition* for the trigger is that the key of the triggering tuple is not in the state relation of the corresponding conceptual value node and that the condition of the conceptual value rule are true. The *action* of the trigger is to insert the key of the triggering tuple into the state relation of the conceptual value node.

Place another ECA trigger into the database relation upon which the conceptual rule's conditions are defined. The *event* is DELETE or

---

[17]The translation of an arbitrary propositional expression into disjunctive normal form (DNF) is straightforward.

UPDATE. The *condition* for the trigger that the key of the triggering tuple is in the state relation for the corresponding conceptual value node and the negation of the condition of the conceptual value rule is true. The *action* of the trigger is to delete the key of the triggering tuple from the state relation of the conceptual value node.

Place a trigger on the *event* INSERT into the conceptual value node's state relation. The *condition* is **true** and the *action* is to insert the static key of the of the inserted tuple into the state relations of all the *children* of the conceptual value node for which the static key is not already present.

Place a trigger on the *event* DELETE into the conceptual value node's state relation. The *condition* is that the static key of the deleted tuple is not found in any other tuples across the conceptual value node's state relation and the *action* is to delete the static key of the of the deleted tuple from the state relations of all the *children* of the conceptual value node for which the static key is already present.

(6) For each high-level concept node:

Place a trigger on the *event* INSERT into the high-level concept node's state relation. The *condition* is that the inserted key should satisfy the DNF parent expression. If it does then the *action* is to insert the key into the state relations of all the *children* of the high-level concept node for which the static key is not already present. If the *condition* is not true then abort the insert into the node's state relation.

Place a trigger on the *event* DELETE into the high-level concept node's state relation. The *condition* is that the deleted key should not satisfy the DNF parent expression. If it does not then the *action* is to delete the key from the state relations of all the *children* of the high-level concept node for which the static key is already present. If the *condition* is not true then abort the deletion from the node's state relation.

(7) For each action node:

Place a trigger on the *event* INSERT into the action node's state relation. The *condition* is that the inserted key should satisfy the DNF

parent expression. If it does then the *action* is to generate a notification message with explanation and send this message to the appropriate person.

Place a trigger on the *event* DELETE into the action node's state relation. The *condition* is that the inserted key should not satisfy the DNF parent expression. If it does not then the *action* is to generate a notification message with an explanation and send this message to the appropriate person.

6.3.3 Example Hierarchy

Using this algorithm, the action rules, high-level concept rules, and the conceptual value knowledge of the scenario, form the action Inference Hierarchy in figure 1.

This diagram shows the class node identifiers and hyper-edges expressing the DNF dependency relations for the action inference hierarchy.

Explanations over the action Inference Hierarchy are computed through applying the algorithms discussed earlier in this paper. The state relations in the nodes determine if an airport is a member of the class associated with a node.

6.4 Example Reports and Generated Messages

We give here an example of this system by presenting a set of static airport tuples and a set of reports that are released on airports and enemy positions. Because there might be multiple reasons why a message is sent, a recipient might request an explanation of *why* they received a message. Such explanations may also be embedded in the message. In the examples here, these explanations are provided in the message.

Consider that there are three airports tuples in the database:

```
AIRPORTS( 1, Tobruk,      30 N, 15 E)
AIRPORTS( 2, El Alamein,  29 N, 25 E)
AIRPORTS( 3, Alexandria,  28 N, 31 E)
```

This scenario starts and the reports that are:

```
AIRPORT_WEATHER_RPT(1, 10 km/hour, 76 F, 0'' rain)
AIRPORT_MAINTENANCE_RPT(1, 0 planes)
AIRPORT_WEATHER_RPT(3, 44 km/hour, 56 F, 4'' rain)
```
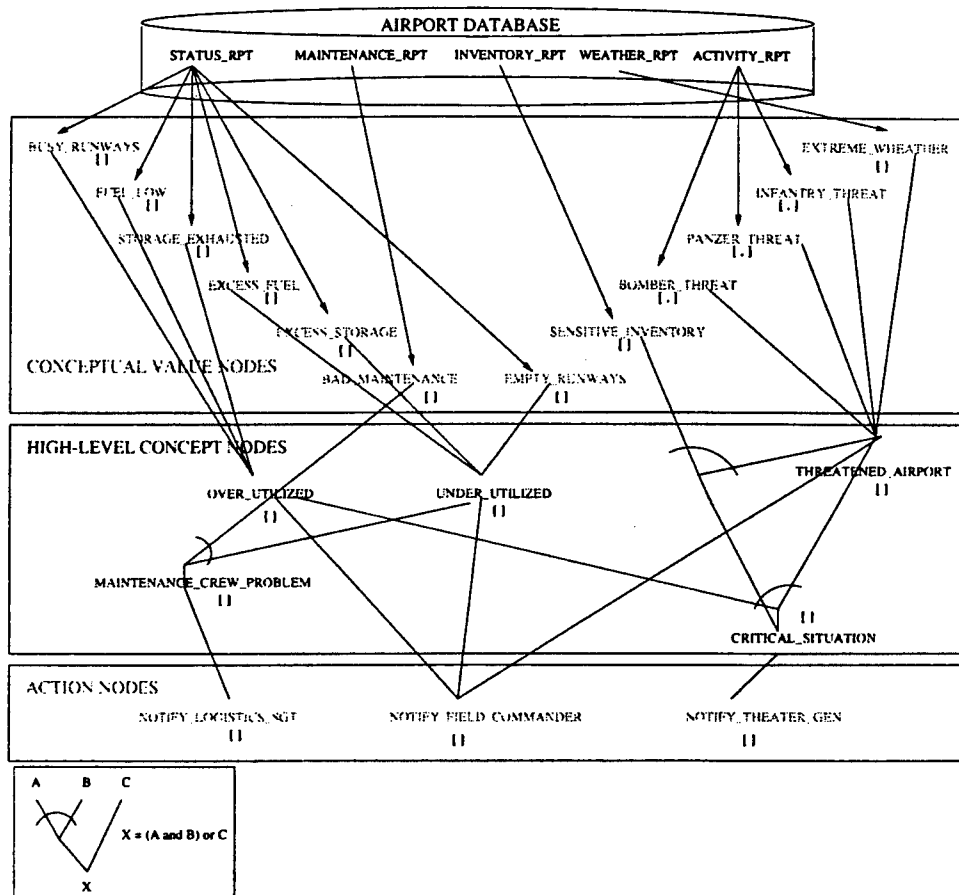
Figure 1: Action Inference Hierarchy for Airport Scenario.

To Field Commander Eisenhower: ``Airport at Alexandria is
 threatened because of wind speeds greater than 25km/hour.''

AIRPORT_STATUS_RPT(1, 3 rws, 90,000 storage)
AIRPORT_STATUS_RPT(2, 3 rws, 40,000 storage)
AIRPORT_INVENTORY_RPT(1, cipher machin: ..
ENEMY_ACTIVITY_RPT(1, Infantry, 28 N, 12E)

To Field Commander Ritchie: ``Airport at Tobruk is threatened
 because German infantry force 1 is near by.''

```
To General Montgomery: ''Airport at Tobruk is in a critical
  situation  because German infantry force 1 is near by and
  there is a cipher machine on site.''

AIRPORT_MAINTENANCE_RPT(2, 7 planes)

To Officer Walker: ''Airport at El Alamein has a MAINTENANCE
  CREW problem because the number of disabled aircraft is 'high'
  and the Airport at El Alamein is not low on fuel and does not
  have busy runways and has not exhausted its storage space.''

ENEMY_ACTIVITY_RPT(2, Panzer, 27 N, 23 E)

To Field Commander Ritchie: ''Airport at El Alamein is
  threatened because German Panzer force 2 is near by.''
```

As an example of negative conditional explanation, General Montgomery might ask why the airport at El Alamein is not in a critical situation given that it is in a threatened situation: The answer would be:

```
'' The airport at El Alamein is not in a critical situation,
   Because there is no sensitive inventory on site.''
```

This technology could be integrated into a GIS. In such a system, airports would indicate their status[18] either through text annotations (as they are here) or through color or other visualization techniques. The user would be permitted to click on an airport icon and receive an explanation of why it has the status it does. An explanation for why it has the status would then be presented in a pop up window. Alternatively there would be a way to let users request explanations for why an airport did not have a particular status.

7   Comparison to Previous Work

As stated previously, there have been attempts to endow intelligent system with explanatory capabilities[11][9][10]. Although such systems demonstrated the usefulness of explanation, there was little in the way of formal justification of the completeness and correctness of explanation. Though

---

[18]That is which classes in the action inference hierarchy it is a member of.

the Inference Hierarchy is a simple computational mechanism, it does offer complete and correct explanation of instance classification. In fact, because of the undecidability problems for Turing equivalent mechanisms, complete explanation requires computational simplicity.

Work in abduction[2] and in explanation-based learning[6][7] is mainly focused on finding most likely explanations of exogenous phenomena. In such systems the information or knowledge provided for a problem is incomplete, and the system is required to provide a hypothesis that best explains the observed information. In the case of explanation-based learning, explanations are generalized and used to repair the knowledge structures employed in problem solving. In contrast, the work here is focused on the problem of providing complete and correct explanations to external audiences of a simple, modular computation system - the Inference Hierarchy.

The Inference Hierarchy mechanism employing conjunction and disjunction is similar to the instance recognition problem in description logics[1][5]. Such knowledge representation systems employ conjunction and disjunction as well as other concept restrictions. However the Inference Hierarchy does not presume that it is representing concepts and may be used to represent arbitrary states in a computation. Moreover this work treats the problem of computing explanation of instance membership over such hierarchies whereas description logics are mostly concerned with computing the subsumption partial ordering among defined concepts. Notably in this work the question of negative instance membership along with conditional explanation is treated.

This work relates to work in generating explanations for misconceptions in database queries[3]. In DATALOG, queries may involve view predicates, predicates recursively defined over others. Furthermore, a predicate may be defined by several clauses, which adds disjunction. So a query's derivation tree is an AND/OR tree. A query is a complex misconception if there is no path by which it may have answers. An explanation of a complex misconception then must account for the possible derivation paths. This relates to some of what is being explored here, especially in the more general case of taxonomies with-negation and disjunction.

The Inference Hierarchy is applied to the problem of active mediation of relational databases. Through the use of simple ECA triggers and the addition of $O(n)$ temporary tables, the Inference Hierarchy may be integrated with a relational database system. This extends the space of scalable active database approaches that leverage off of simple ECA triggers on relations. The explanation algorithms give more than a simple enumeration of the rules matched; it gives minimal sets of matching conditions, as well

as negative and conditional explanation.

8 Discussion

An explanation should be concise and, in itself, useful to a user. Conditional explanation promotes concision by avoiding explanations that include what the user is already aware of. To consider the question of what is useful to the user, we must consider why an explanation is being sought.

An explanation is requested either because the user is curious about why an inference is or is not made, or because the user wishes to perform a corrective action to lead to, or away from, an inference. Let us term the former type of explanation *probative* and the later type of explanation *corrective*.

As an example of probative explanation, suppose a commander is surprised to hear that there is sensitive inventory at a sight that had previously been considered inconsequential. The commander seeks an explanation to get a more complete understanding of the situation. This works in the negative case as well. The commander may be surprised to find out that a facility is not considered threatened. Reporting that the facility has a special defensive barrier is an explanation. For probative explanation it is useful to supply a series of alternative explanations if they exists. Based on economy of communication, and given no other information, a good heuristic is to present those explanations with the fewest number of conditions first. In general the algorithms in this paper enable practical generation of the k minimal explanations.

As an example of corrective explanation, suppose a commander hears that one of his airports is threatened and he would like to alleviate the situation. This is the positive case of corrective explanation where a set of conditions that, if reversed, would lead to non-membership in the class. For example, to make the airport at Tobruk not threatened, neutralize Nazi force 1. In the negative case, the commander is presented with conditions that if reversed would establish membership in the class. For example To make the facility operational, supply it with 7,300 gallons of gas and a new barracks to sleep 120 soldiers. Corrective explanation is different than probative explanation for two reasons. First the conditions in a corrective explanation should be weighted based on the listeners ability to reverse the conditions. This adds additional representation requirements and complicates algorithms for finding globally minimal explanations. Second for a positive corrective explanation to be useful in itself, it must include all sup-

porting inference paths. That is all supporting inference paths that lead to the instance being inferred to be a member of the class must be blocked. Interestingly corrective explanation in the negative case does not require this. Only one path of inference that the user can open suffices[19] Special purpose algorithms that seek to provide a minimal set of minimal explanations that block all paths of inference may serve as the basis of providing positive corrective explanations. In the simplest case, this will consist of a single condition on the class in question. In the most complex case, the set of blocking conditions will have to be identified.

9 Conclusion

$O(c)$ time algorithms are given to compute positive, negative, and conditional explanation over an Inference Hierarchy limited to conjunctive parents, where c is the total number of conditions in the Inference Hierarchy. Algorithms requiring $O(c^2k)$ time and space are given for computing the k-minimal positive, negative and conditional explanations over disjunctive hierarchies with independent conditions. Worst-case exponential time is required for computing negative and conditional negative explanations over disjunctive hierarchies with dependent conditions. All types of explanation are worst-case exponential time for the k-minimal explanations over hierarchies using negation in parent relationships. The Inference Hierarchy is applied to the problem of active mediation of relational databases. Inference hierarchy algorithms are implemented through triggers and temporary relations.

10 Bibliography

References

[1] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

---

[19]Of course in Inference Hierarchies that include negation, we are just one step way from chai ging signs, so negative corrective explanation may carry the same requirement as positive corrective explanation.

[2] Gerhard Brewka and Kurt Konolige. An abductive framework for generalized logic programs and other nonmonotonic systems. In *IJCAI-93*, 1993.

[3] Parke Godfrey. Minimization in cooperative response to failing database queries. Technical Report CS-TR-3348, University of Maryland Institute for Advanced Computer Studies Dept. of Computer Science, Univ. of Maryland, College Park, MD, September 1994.

[4] W. Kim, editor. *Modern Database Systems: The Object Model, Interoperability and Beyond.* Addison-Wesley, Reading, Massachusetts, 1994.

[5] R. MacGregor and M. Burstein. Using a descriptive classifier to enhance knowledge representation. *IEEE Expert*, 6(3):41–47, 1991.

[6] Tom Mitchell, R. M. Keller, and S. Kedar-Cabelli. Explanation-based generalization: a unifying view. Technical Report ML-TR-2, SUNJ Rutgers U, 1985.

[7] Michael Pazzani, Michael Dyer, and Margot Flowers. Using prior learning to facilitate the learning of new causal theories. In John McDermott, editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 277–279, Milan, Italy, August 1987. Morgan Kaufmann.

[8] R. Reiter. *Logic and Databases*, chapter On Closed World Databases. Plenum Press, 1978.

[9] E. Shortliffe. *Computer Based Medical Consultations:MYCIN.* Elsevier North Holland Inc., 1976.

[10] W. Swartout, C. Paris, and J. Moore. Design for explainable expert systems. *IEEE Expert*, 6(3):58–64, 1991.

[11] T. Winograd. *Understanding Natural Language.* Academic Press, New York, 1972.

Query Formulation from High-level Concepts with Multi-modal Interface

Guogen Zhang, Wesley W. Chu, Gladys Kong, and Frank Meng

Phone: (310)825-2047

Fax: (310)825-7578

Email: wwc@cs.ucla.edu

October 29, 1998

Abstract

A new approach for query formulation based on a semantic graph model is presented, which provides a semantic representation of the data in the database augmented with user-defined relationships. The graph model can be semi-automatically generated from the database schema. The query formulator allows users to specify their request and constraints in high-level concepts to formulate queries instead of using a database query language. The query formulator completes a query based on the user input and ranks the formulated query candidates according to the probabilistic information measure. English-like query descriptions can also be provided for the user to resolve ambiguity when multiple queries are formulated from a user input. Further, the system allows the user to interact with the system and the user can select the desired query. A prototype system using the proposed technology with a multimodal interface consisting of GUI and voice interface has been implemented at UCLA. The formulator is operating on top of the cooperative database system (CoBase) to formulate SQL queries.

1 Introduction

Most database interfaces provide poor guidance for ad-hoc query formulation which burdens the user to learn or to know precisely the query language and the database schema. An ideal query interface should assume that users may have little technical knowledge and possibly possess no knowledge concerning the schema of the database. Many current and future applications of DBMSs, e.g., scientific computing and decision support, require user interaction based on many

ad-hoc queries, instead of the conventional invocations of pre-compiled and stored application programs. As database schemas become larger and more complex, there is a need to develop an intelligent, high-level query interface to allow users to specify queries by high-level concepts and constraints.

The universal relation model [Ull88, Var88] based on the uniqueness assumption of relationships attempts to relieve users of the burden of specifying joins. But it does not allow arbitrary user-defined concepts in its model, thus limiting its applicability [Cod88]. The maximal object theory used to derive tree schemas from a cyclic schema also has limited applicability.

Wald and Sorenson [WS84] formalized the query completion problem as a Steiner Tree problem, and presented a search algorithm for partial 2-tree graphs. They used a deterministic directed costs for the edges, such as the cardinality of relationships, to measure the complexity of queries. Ioannidis and Lashkari [IL94] considered path expression completion in object-oriented queries with the partial order relationship between different paths for ranking. All these query formulation methods can only generate simple queries. Therefore, we propose to use a semantic query graph which represents a more general approach in query formulation

The semantic graph, which models the objects in the database and user-defined relationships, can be semi-automatically generated from a database schema with user-defined relationships augmented by domain knowledge. Based on graph search methods, queries can be formulated by finding a path in the semantic graph which encompasses the query input given by the user. Since it is possible to have multiple paths given a set of partial input information, the system ranks the multiple paths based on the amount of information present in the nodes and links of the path.

Our proposed query formulation technique has several unique features. First, it allows user-defined relationships incorporated into the graph. Second, it does not have specific limitations on the graph structure, as required in [WS84]. Third, a probabilistic information measure is used for query ranking. Finally, English-like query descriptions can be generated for resolving query ambiguity.

The rest of the paper is organized as follows. A discussion of the semantic graph is given in section 2. Section 3 describes query formulation from high-level concepts, and section 4 describes a multimodal user interface to the system. Implementation and experience are presented in Section 5. Finally, a conclusion is presented in section 6.

2 The Semantic Graph Model

In addition to the typical constructs used to represent entities and relationships in a semantic model, our semantic graph model contains more information to support user query interfaces and query formulation mechanisms. The basic constructs of our semantic model are the following: (1) strong and weak entity types; (2) ISA relationship between entities; (3) HAS relationship between a strong and a weak entity; (4) Simple link between entities without its own properties; (5) complex associations between entities with its own properteis; and (6) User-defined relationships between entities.

To allow semi-automatic generation of semantic graphs from relational schemas and to simplify the information carried by links. For relational databases, nodes are used to represent relations and links are used to represent joins.

Formally, a *semantic graph* for a relational database is a weighted undirected graph $G = (V, E)$, where each node in the set of nodes V corresponds to a relation, and each link in the set of links E corresponds to a join between relations of the link's two end nodes. The joins can be natural or user-defined. Associated with each node and link is a conceptual term that can be used by the user to refer to the corresponding elements in the graph. Weights are assigned to the nodes and links in accordance with their relative importance, which are used in the query formulation. Lexical information is used to support English-like query descriptions.

Part of the transportation database semantic graph is shown in Figure 1. It contains information about airports, aircrafts, seaports, channels, ships, etc. The link (AIRPORTS) HAVE (RUNWAYS) is an example of a natural join link, while the link (AIRCRAFTS with AIRFIELD_CHARS) CAN LAND (on RUNWAYS) is an example of a user-defined link. There are no self-join links in this graph. Note that the weight of each link is shown along the link, which gives a relative measure of the specificity of the link.

2.1 Semantics of Subgraphs

A link of the semantic graph represents a join. In general, a connected subgraph represents a relational algebraic expression consisting of a set of joins represented by the links. We call a connected subgraph a *query topic* as shown in Figure 2.

For example, in the transportation semantic graph, "CAN LAND" is a link between AIRCRAFT_AIRFIELD_CHARS and RUNWAYS. It corresponds to a complex join condition:
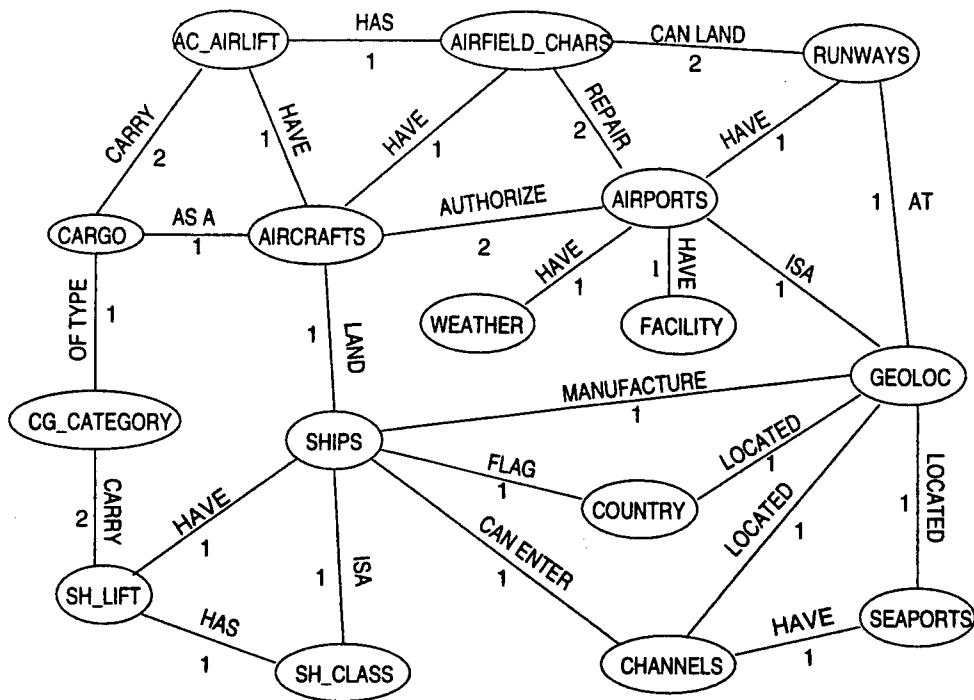
Figure 1: Semantic graph for the transportation database.

```
AIRCRAFT_AIRFIELD_CHARS.PT_MIN_AVG_LAND_DIST_FT <=
    RUNWAYS.RUNWAY_LENGTH_FT
AND AIRCRAFT_AIRFIELD_CHARS.PT_MIN_RUNWAY_WIDTH_FT <=
    RUNWAYS.RUNWAY_WIDTH_FT
```

The result of the algebraic expression evaluated against the database will be all the tuples containing information of an aircraft type and a runway such that the aircraft satisfies the runway length and width for landing.

When an algebraic query expression corresponding to a query topic is evaluated against the database, each of the resulting tuples will contain information about the airfield characteristics of an aircraft type, a runway, its airport, its geographical location information, and its country information, such that an aircraft of the aircraft type can land on the runway of the airport at the geographical location of the country (see Figure 2).

Users are usually only interested in a subset of the tuples generated from the evaluation of a query topic by imposing selection conditions (i.e. constraints) on the topic. For example, the user can specify the aircraft type to be "C-5", and the country to be "Tunisia". We call these *query constraints* on the query topic.
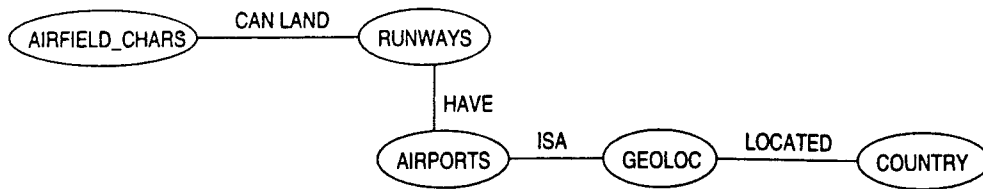
Figure 2: A query topic for "aircraft can land on airports at geographical locations of countries".

A *query aspect* of a query topic is a list of attributes (and their expressions) that are contained in the nodes of the query topic. A query aspect corresponds to the select list of an SQL query or the projection part of an algebraic expression. The query topic, constraints, and aspect together can be converted into an algebraic query expression.

2.2 Semi-automatic Generation of Semantic Graph

In current database systems, after a semantic model is converted into relations, it is no longer stored in the data model (i.e. the schema). Thus we have to use reverse engineering[CBS94] to reconstruct the semantic model from a database schema. However, the mapping from a relational model to the extended entity-relationship (EER) model [TYF86] is not one-to-one. Therefore, in general human intervention is needed to construct the semantic model from the corresponding database schema.

With our simplified definition, an initial semantic graph model can be automatically generated based on all the natural join links between relations. User-defined links can then be added. Other information, such as conceptual terms, can be assigned to the nodes and links.

A natural join between two relations usually represents a relationship through a key and a foreign key[1]. If the same names are used for attributes referring to the same domain, natural join links between relations in a database schema can be automatically generated based on the corresponding key attribute names. However, the same attribute name may refer to different domains, and two different attributes may refer to the same domain. To accommodate this situation, when checking if a key of a relation appears in another relation as a foreign key, we use domain names instead of attribute names. The domain names have to be provided by the designers.

To allow self-joins to the same relation, a relation is duplicated if the key is used within the

---

[1]To avoid an excessive number of links being generated, the links are limited to natural joins between a key and a foreign key and a link is not generated for a join between two foreign keys.

same relation. This will ensure that one of the self-join relationships can be found. If more than one such recursive relationship exists, manual editing is required to add more links.

To generate all the links with natural joins as their conditions, we find all the natural joins between pairs of nodes for a given array of relation nodes. The time complexity of the algorithm is $O(kmn^2)$, where k is the maximum number of keys in a relation, m is the maximum number of attributes in a relation, and n is the number of relations, including the duplicates.

For example, consider the following three relations:

```
AIRPORTS( APORT_NM, ELEV_FT, GEOLOC_TYPE, GLC_CD, HARDSTANDS,
          HNS_SORTIES, MAX_SORTIES, MILITARY_CIVILIAN_FLAG,
          PARKING_SQ_FT, POL_BBLS, SECONDARY_NM, STATUS_FLAG ),
     KEY ( APORT_NM )


RUNWAYS( APORT_NM, GLC_CD, RUNWAY_LENGTH_FT, RUNWAY_NM,
         RUNWAY_WIDTH_FT, SURFACE_FLAG ),
     KEY ( APORT_NM, RUNWAY_NM )


GEOLOC( CIVIL_AVIATION_CD, CY_CD, GLC_CD, GLC_LNCN, GLC_LOG_RGN_CD,
        GLC_LTCN, GLC_NM, GSA_CITY_CD, GSA_COUNTY_CD, GSA_STATE_CD,
        INSTLN_TYP_CD, LATITUDE, LONGITUDE, PRIME_GLC_CD,
        PROVINCE_CD, RECORD_OWNER_UIC ),
     KEY ( GLC_CD )
```

According to the key and foreign key relationships, we can find the following three natural join links.

1. AIRPORTS and RUNWAYS, linked by APORT_NM.

2. AIRPORTS and GEOLOC, linked by GLC_CD.

3. RUNWAYS and GEOLOC, linked by GLC_CD.

Since GLC_CD is a foreign key in both AIRPORTS and RUNWAYS, a link between these two relations by GLC_CD is not generated.

2.3 Weights of Nodes and Links

In the query formulation, multiple queries may be formed for a given user input. The *information* [Qui93] of nodes and links is used (expressed in weights) for selecting and ranking query candidates.

The *weight* for an element e (a node or a link) in a semantic graph measures the *information content* of e; that is,

$$I(e) = -\log P(e)$$

where log denotes the base 2 logarithm, and $P(e)$ is the probability of using e in queries.

The definition is consistent with that used in information theory which represents the number of bits needed to encode a node or link. The measure reflects the information content of a node or link. A smaller value of $I(e)$ means a larger $P(e)$, thus e will more likely appear in queries.

For simplicity in computing the information of a subgraph, we assume that all the nodes and links are independent. For a subgraph with a set of elements (nodes and links) $E = \{e_i | i = 1, \ldots, n\}$, the independence assumption implies that the information measure is additive, that is,

$$P(E) = P(\{e_i | i = 1, \ldots, n\}) = \prod_{i=1}^{n} P(e_i).$$

Thus

$$I(E) = -\log P(E) = -\log(\prod_{i=1}^{n} P(e_i)) = \sum_{i=1}^{n} I(e_i) \tag{1}$$

Weight Update   The weights for nodes and links can be approximated by frequency and updated by counting. Let c_i be the number of times that e_i is used in queries, and c be the total number for all the elements used in a set of queries, then

$$P(e_i) = \frac{c_i}{c} \tag{2}$$

When element e_i is used in a new query, the weights can be updated by incrementing its corresponding count and the total count.

Initial Weight Assignment   If a large collection of queries are available at the beginning, initial counting can be performed. But if the query set is not available or is too small to be statistically significant, we can assign an equal initial weight to all the nodes and assign weights to links based on the link types and their specificity. An example of link types and their sample weights is shown in Table 1. Based on the semantics of the links, in general, the relationships among the weights are $0 < w_1 \leq w_2 \leq w_3 \leq w_4 \leq w_5 \leq w_6$.

Link Type	Weight
specific-entity ISA generic-entity	$w_1 = 1$
strong-entity HAS (PROPERTY) weak-entity	$w_2 = 2$
entity ROLEOF association	$w_3 = 2$
entity LINK entity	$w_4 = 3$
entity USER-DEFINED-ROLEOF association	$w_5 = 3$
entity USER-DEFINED-LINK entity	$w_6 = 4$

Table 1: Link types and their corresponding weights in the semantic graph. The numbers are the sample weights.

Once the initial weights are assigned, they are normalized according to the probability property (summed to 1), and then converted into initial counts.

3   Formulation of Simple Queries from High-level Concepts

To formulate simple queries without detailed knowledge of a query language or the database schema, our query formulator only requires users to specify concepts, attributes, and values about a query. Based on these input, the system constructs the query via the semantic graph.

The query topic is the major source of complexity in formulating a query. The user input contains unconnected nodes and links for constructing a subgraph for the query topic. To formulate a query, we need to add links and nodes to extend into a subgraph for a query topic.

When the graph is cyclic, multiple links can be connected for the same set of nodes which can cause query ambiguity. We resolve this problem by (1) ranking the candidate queries based on their information measure, and (2) generating English-like query descriptions for the candidate queries to allow the user to select the desired one.

3.1   An Example

The user input consists of three parts: the query aspect which corresponds to the SELECT clause of an SQL query, the constraints, and special link requirements expressed in conceptual terms. The user interface then converts the user input into relations, attributes, and links. Consider the formulation of the query *"Find airports in Tunisia that can land a C-5 cargo plane."* The user input is as follows (see also Figure 6):

- query aspect: `AIRPORTS.APORT_NM`;

- constraints: `AIRCRAFT_AIRFIELD_CHARS.AC_TYPE_NAME = 'C-5'` and
  `COUNTRY_STATE.CY_NM = 'TUNISIA'`;

- links: "CAN LAND".

The query formulator searches the transportation semantic graph (Figure 1) and adds missing links and nodes to complete a subgraph. A list of query candidates can be formulated. The following is the first query candidate[2]:

```
SELECT  R3.APORT_NM
FROM    AIRCRAFT_AIRFIELD_CHARS R0, AIRPORTS R3,
        COUNTRY_STATE R10, GEOLOC R11, RUNWAYS R16
WHERE   R0.AC_TYPE_NAME = 'C-5'
        AND R10.CY_NM = 'TUNISIA'
        AND R0.WT_MIN_AVG_LAND_DIST_FT <= R16.RUNWAY_LENGTH_FT
        AND R0.WT_MIN_RUNWAY_WIDTH_FT <= R16.RUNWAY_WIDTH_FT
        AND R11.GLC_CD = R3.GLC_CD
        AND R3.APORT_NM = R16.APORT_NM
        AND R10.CY_CD = R11.CY_CD
```

3.2 Query Formulation as a Graph Search Problem

Given a user input for query formulation, we can process it into an incomplete query topic T_I, an aspect A, and a constraint set S. T_I contains the links specified in the user input, the nodes involved in the links, and the nodes in A and S.

Since T_I is not usually a connected subgraph, we need to choose additional links and relevant nodes from the semantic graph to extend T_I to form a connected subgraph for the query topic. We call these links and nodes a *query completion candidate* for T_I.

Property of a query completion candidate *Given a semantic graph $G = (V, E)$, to formulate a query from an incomplete input query topic $T_I = (V_I, E_I)$, where $V_I \subseteq V$ and $E_I \subseteq E$, is to find a query completion candidate $T_C = (V_C, E_C)$ for T_I such that query topic $T = T_I \cup T_C = (V_I \cup V_C, E_I \cup E_C)$ is a connected subgraph of G, where $V_C \subseteq V$, $E_C \subseteq E$, $V_C \cap V_I = \emptyset$, and $E_C \cap E_I = \emptyset$,*

---

[2]The aliases for the relations in the query are node IDs in the semantic graph.

If the semantic graph is cyclic, there can exist more than one query completion candidate for the same input. We propose to use the following minimum missing information principle for ranking the candidates.

Minimum Missing Information (MMI) Principle *The query completion candidate T_C (the missing links and nodes) for an incomplete input topic T_I contains the minimum information; i.e.* $\min I(T_C)$.

Based on equation 1, $I(T_C)$ can be computed from the information (weight) of the nodes and links as follows.

$$I(T_C) = \sum_{v \in V_C} I(v) + \sum_{e \in E_C} I(e) \tag{3}$$

Thus the MMI principle provides us the measure for ranking the query completion candidates.

The smaller the $I(T_C)$, the more likely the completion candidate will meet the user's query intention. Links and nodes of smaller weights have a higher probability of being used in queries. Therefore they are the more likely candidates to be the intended query. Due to the independence assumption, the probability value used in the ranking is an approximation. Thus we use a set of completion candidates and let the user select one.

Based on the MMI principle, the end points in the completed subgraph are from the user input. Deleting any end node that is not in the input will reduce the information of the completion candidate without affecting the connectivity. Thus, our MMI principle is consistent with the formulation of the query completion problem as a Steiner Tree problem [WS84]. According to [WS84, HR92], query completion as a graph search problem is NP-complete.

3.3 Algorithm for Searching Query Completion Candidates

A user input contains a query aspect, a constraint set, and a link set. The required pre-processing for the input is to extract all the nodes that appear in the aspect and constraints. The nodes together with links from the input form the incomplete input query topic T_I for searching the query completion candidate T_C.

The high-level query formulation procedure is illustrated in Figure 3. The incomplete input topic is decomposed into a set of connected *components*. The process of finding a completion candidate is to repeatedly connect two components via a path and form into a larger component. This merging process terminates until reduced to a single component. We use the following two methods to limit the search scope:

- L-step-bound paths: paths that connect two components bounded by at most l links. This confines the search to a small area surrounding the input subgraph.

- K-minimum completion candidates: only a maximum of k candidates with minimum weights are used in the search process. This further trims the search within the set of candidate paths.

L-step-bound Paths L-step-bound paths are used to limit the search within the neighborhood of the input subgraph. This is based on the observation that each query topic will only span a small region of the semantic graph. L-step bound paths eliminate the long paths for connecting components. L-step-bound paths are found through a breadth-first search. Paths with l links that have not yet reached a destination node are dropped.

The worst-case time complexity for finding l-step-bound paths with n components is $O(n^2 ma^l)$, where m is the maximum number of nodes in a component and a is the maximum number of links connected to a node.

K-minimum Completion Candidates During the search for completion candidates, α-β pruning is used to trim the search branches. If a partial completion has a larger weight than the maximum weight of the current k-minimum completion candidates, this partial completion is excluded from further search.

For the current component list, the algorithm repeats the following process until no more paths can be tested:

1. Find l-step bound paths for the current component list

2. For each path, merge the two components connected by the path

3. Check if only one component remains in the list

4. If yes, a completion candidate is found, and go test the next path

5. Otherwise, check if the current partial completion candidate is acceptable for the k-minimum completions;

6. If yes, go forward with the resultant component list;
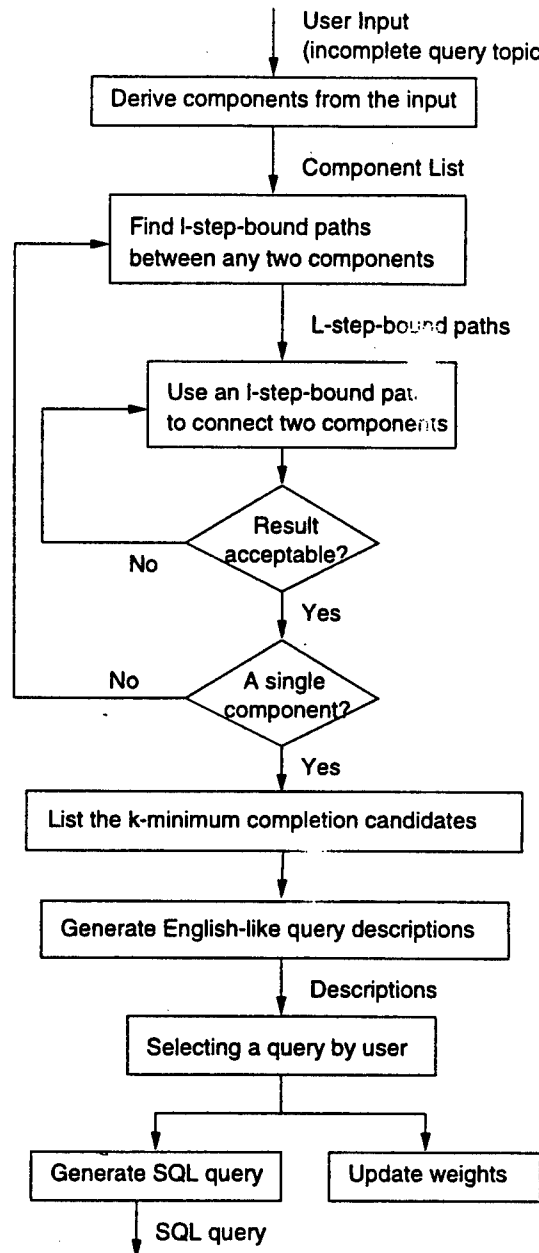
Figure 3: The flow diagram of the query completion procedure.

	REPAIR
(1)	AIRFIELD_CHARS ——————— AIRPORTS
	2

(a) Initial components (b) 2-step-bound paths
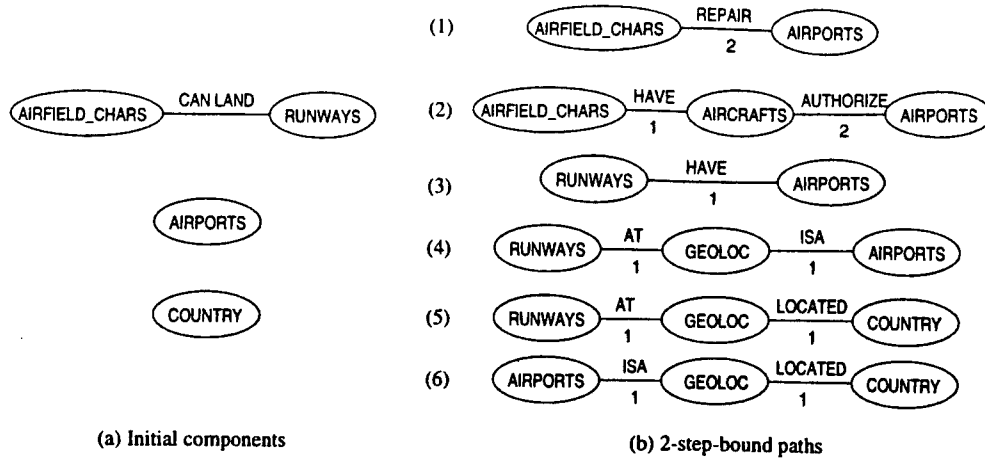
Figure 4: Initial components and 2-step-bound paths for the "CAN LAND" query.

7. If not, test the next path. If no more paths left, go back use previous component list (backtracking).

A larger l will be used if a completion candidate cannot be found for the current l.

The search problem is NP-complete. The worst-case time complexity can grow exponentially with the size of the graph (assuming $P \neq NP$). In most cases, however, the search will touch only part of the graph.

For the example query in Section 3.1, we have the initial components and 2-step-bound paths as in Figure 4. We found 2-minimum completion candidates based on the 2-step-bound paths. The first contains paths (3) and (6) with an additional node GEOLOC, and the second contains paths (3) and (5) with the same additional node GEOLOC. After converting to SQL, the first completion condidate yields the query as shown in Section 3.1.

Correctness of the Query Completion Algorithm The k-minimum completion algorithm performs an exhaustive search with α-β pruning, which is equivalent to the exhaustive search. The query completion algorithm will find k-minimum completion candidates, as long as the algorithm for l-step-bound paths does not miss any paths that qualify in the k-minimum completion candidates.

Using l-minimum weight paths requires a depth-first search in the semantic graph which is not locally bounded. Therefore, l-step-bound paths are used which requires only a breadth-first search with a definite bound. For a reasonably large l, l-step-bound paths yield all possibly qualifying paths and ensure obtaining k-minimum completion candidates.

118
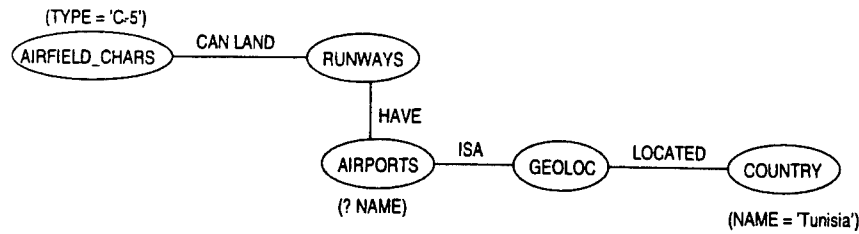
Figure 5: The "CAN LAND" query in the graph form for English-like query description.

3.4 Generating English-like Query Descriptions for Disambiguation

When there are multiple sets of links connecting the components of the incomplete input query topic, multiple queries can be generated which yield different answers, although sometimes they may be equivalent. Thus our system provides English-like query descriptions for the user to select one of the candidate queries that satisfies his query goal.

English-like query descriptions cannot be easily generated from SQL queries because of the lack of semantic information. However, the query representation based on the semantic graph can be used to provide such semantics. Entity nodes are translated into nouns, and links are translated into verbs. Nodes for complex associations including links can also be translated into verbs. Selection conditions on entity nodes are translated into adjective phrases (modifiers on nouns), whereas conditions on association nodes are translated into adverbial phrases [Che83].

Lexical formulas are associated with the graph elements. Domain-specific conceptual terms assigned to relations, attributes, and links can be utilized for the description. The following is the algorithm to generate English-like query descriptions.

- Translate the select list items of a relation into "*attribute1, attribute2, ...* of *relation*";

- Use one sentence for each link in the query topic;

- Put the selection constraints into corresponding sentences as modifiers.

- List the sentences involving relations in the query aspect first.

We use the above algorithm to generate an English-like description for the query example in Section 3.1. The query in a graph form including the topic, the constraints, and the aspect is shown in Figure 5. Its literal translation into English-like sentences is as follows:

"*Find NAME of AIRPORTS, WHERE:*

AIRPORTS is a GEOLOC;

119

GEOLOC is located in COUNTRY with NAME Tunisia;

AIRPORTS have RUNWAYS;

AIRCRAFTS with TYPE NAME C-5 can land on RUNWAYS."

3.5 Capabilities of the Query Formulator

To formulate queries containing aggregate functions, a default GROUP BY clause is generated for queries by putting all the non-aggregate function expressions in the SELECT list into the GROUP BY clause.

CoBase is a knowledge-based cooperative database system that supports query relaxation to provide approximate query answers if an exact answer is not available [CMB93]. In connection with the CoBase system, we include cooperative operators, such as SIMILAR-TO and NEAR-TO, and relaxation control in our query language CoSQL [CMB93, CYC+96].

Using the proposed query formulation technique, we are able to formulate to the SELECT-PROJECT-JOIN type of queries, with or without aggregate functions, as well as CoSQL queries.

Performance tests were conducted on the transportation database schema for the accuracy and search time on a Sun Ultrasparc II machine. The graph for the transportation database contains 50 nodes and 105 links. For 30 random query samples, all the first query candidate are correct for tree queries. The formulation search time ranges from 84 milliseconds to 871 milliseconds, with an average time of 323 milliseconds, which is well below the query execution time.

4 Multimodal User Interface

The usability of a query formulation system depends heavily on the user interface. Since the goal of the system is to facilitate query formulation, the user must be able to express the criteria of the desired information to the system in an easy and efficient manner. We present a multimodal user interface which accepts input through voice and point-and-click devices (e.g., the mouse).

4.1 Point-and-Click interface

The main goal of query formulation is to ease the process of constructing queries such that users are not required to have detailed knowledge of any query language nor the database schema. The user only has to input the query aspect (the SELECT clause of an SQL query), the constraints, and specific link requirements, then the system attempts to complete the query for the user. A

point-and-click user interface provides a simple and efficient way for users to input the desired information to the system. Also, it allows users to interact with the system during the query formulation process.

To formulate the "CAN LAND" query in example 1, the user has to do the following steps. First the user chooses the relevant subjects from a list of query subjects (e.g. aircraft, airports, seaports, country, ships, etc.) provided by the system. In this example, the subjects: aircraft, airports and country will be selected. The system will then present the user with concepts (tables) under these subjects and a list of attributes for the selected tables. The user has to select the query aspect, which are the attributes he wishes to see in the answer set. Then the user has to specify the attribute values, which corresponds to the query constraints. Based on the partial query input, the system will find a set of relevant user-defined links, such as "can land", "repair", and "authorization". The user-defined links can be added to the semantic graph by domain experts or the user through an interface provided by the graphical user interface. The system presents these links to the user and the user selects the user-defined link that applies. For this example, "can land" would then be selected from the list. Now, the input process is complete and the user can choose to have the system formulate the query.

To formulate a query with the specified concepts, the system needs to link these concepts by JOINs. This corresponds to finding a connected subgraph (the query topic graph) in the semantic graph. In this example, the high-level concept "can land" refers to the rule that specifies the landing requirement of an aircraft:

```
AIRCRAFT_AIRFIELD_CHARS.PT_MIN_AVG_LAND_DIST_FT <= RUNWAYS.RUNWAY_LENGTH_FT
AND AIRCRAFT_AIRFIELD_CHARS.PT_MIN_RUNWAY_WIDTH_FT <= RUNWAYS.RUNWAY_WIDTH_FT.
```

Since multiple subgraphs can be generated based on the same input, it is possible for the system to return multiple completed queries. In order to resolve the ambiguity, the system needs to interact with the user. The system has to present each query candidate as an English-like description to the user to allow him to select the desired query.

The user interface is mainly menu-driven with very minimal typing required (only when the user has to specify a value). This type of point-and-click interface not only reduces the time in entering a query, but also reduces the chance of typing errors.
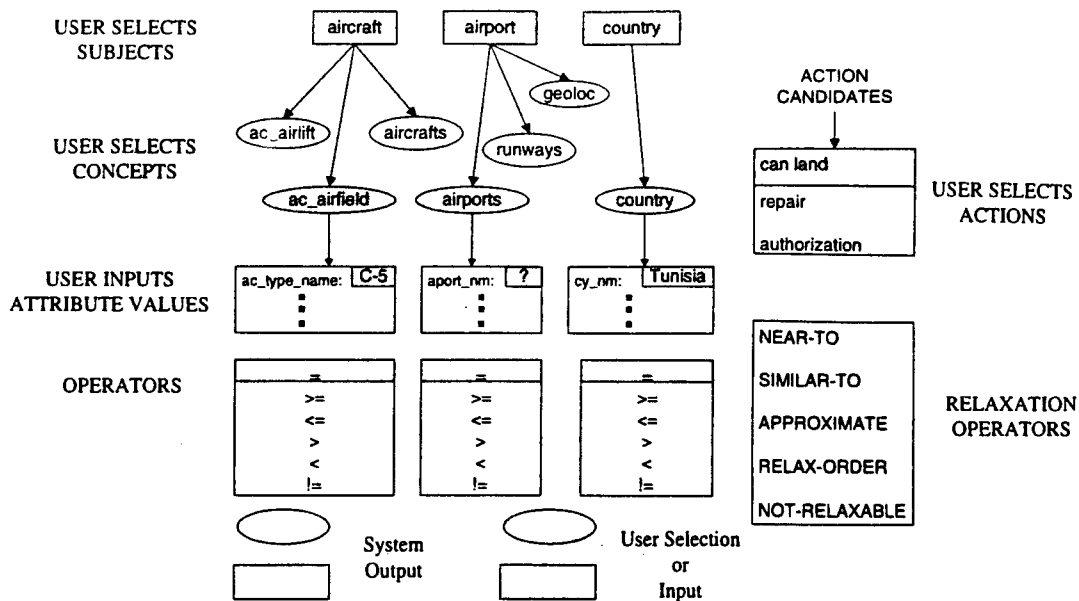
Figure 6: Dialogue between user and system in formulating the query *"Find airports in Tunisia that can land a C-5."*

4.2 CoSQL Operators in Query Constraints

In addition to supporting relational databases, the query formulator is capable of formulating Coopertive SQL (CoSQL) [CYC$^+$96] queries to support cooperative database (CoBase) [CMB93]. Thus, our user interface allows users to specify cooperative operators, such as SIMILAR-TO, NEAR-TO and APPROXIMATE in the query constraints. Users can also specify relaxation control operators such as RELAX- ORDER and NOT-RELAXABLE in their input. For each query constraint, the user has to choose a concept (table), an attribute, an operator ($=, >, <, >=$,$<=$) and/or any CoSQL operators from the system provided menus. Then the user can type in the value for that particular attribute. If the user wants to specify the constraint, runway length greater than approximately 8000 feet, then the user selects the concept RUNWAY, the attribute RUNWAY_LENGTH, the operator $>$, the CoSQL operator APPROXIMATE and then type in the value 8000. If the user wants to further specify this condition as a NOT-RELAXABLE condition, the user simply has to select this condition and click on the NOT-RELAXABLE button to make this condition not relaxable for the CoBase system.

4.3 Map Interface

Another aspect of this point-and-click user interface is that it is integrated with a map server to allow users to specify geographical constraints to the query by drawing a region on the map. It also allows users to visualize the relaxation process and view answers to the query on the map. When the user is interested in asking a map query, he first selects a region on the map, then he goes through the query formulation input process. When the system attempts to complete the query, it will take the geographical constraints specified by the user and based on the context of the user input, it will automatically figure out the appropriate JOINs to add for the geographical conditions. For example, the user can add a geographical constraint to the "CAN LAND" query in example 1. The user may want to find airports that can land a C-5 aircraft only in the southern part of Tunisia. In this case, the user can draw a region in the southern part of Tunisia and then provide the rest of the input to the system as described earlier. Based on the relations specified by the user in the query aspect and constraints (AIRPORTS, AIRCRAFT_AIRFIELD_CHARS and COUNTRY_STATE), the formulator chooses the GEOLOC relation to obtain the latitude and longitude information for the drawn region on the map for this particular query. The necessary JOIN conditions are added in by the formulator and the completed query is submitted to CoBase. If there is no airport in the specified region that can land a C-5 aircraft, the CoBase system will enlarge the region and try to find an airport nearby that satisfies the requirements for landing a C-5. When the system locates such an airport, the relaxed region as well as the airport will be displayed on the map as shown in Figure 7.

4.4 Voice

Voice as an input medium for information systems is useful in environments where the user is unable to conveniently use other input devices such as the keyboard or mouse. Voice will also become more and more useful as computer units become smaller, where a full-sized keyboard is not a feasible input option. Users tend to use natural language when speaking to a computer, and thus, a voice interface must possess some measure of natural language understanding. Our voice interface will limit the domain of natural language that needs to be understood to allow the problem to be more feasible. It will be assumed that the user's requests for information always pertains to the data captured in the database. The goal of the voice system is to capture enough semantics from the user's input to piece together a possible query topic from the semantic graph.
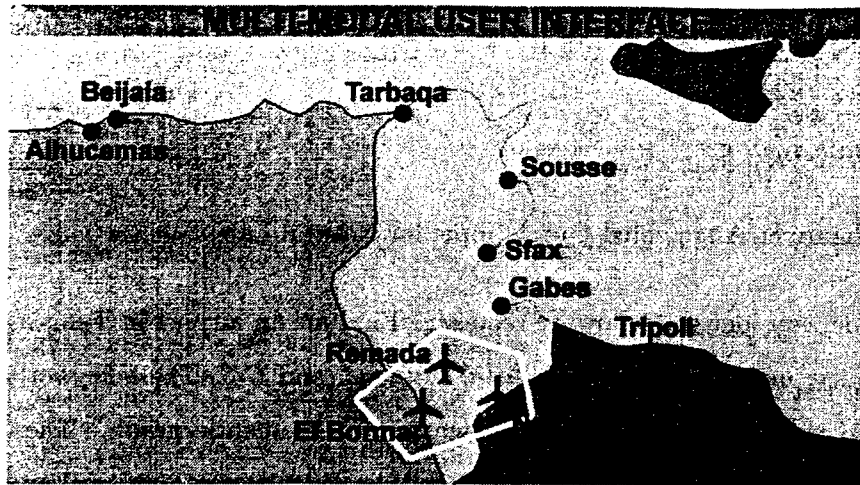
Figure 7: A geo-spatial query "Find airports that can land C-5 in the region specified on the map." The region in the map is sketched by the user. The results (airport locations) are also displayed on the map. Information of a specific airport can be retrieved by clicking on the airport icon.

The techniques used to do this will include keyword spotting and word occurrence statistics based on n-grams. This section will also describe how the semantic graph can be used to produce an English-like response which can be displayed as text or read back using a voice synthesizer.

4.4.1 Using the Semantic Graph for Language Understanding

The meaning of a user's request in a database domain is a query in a formal query language. Since the query formulator needs the query topic and aspect to generate an SQL query, the goal of the language understanding module is to obtain a query topic and aspect from natural language.

To obtain the meaning of the user's request for information, the system searches for keywords which are clues in determining the meaning of the input. In our system, keywords come from two major sources: the semantic graph and the database values. The names of the nodes and attributes in the semantic graph are valuable keywords which represent all the objects within the database domain. Since we are assuming that the user's request pertains to the objects in the database domain, these are the only objects that will be referred to by the user. This limits the *nouns* which are allowed by the system. Other syntactic constructs such as verbs, prepositions, and their corresponding phrases are assumed to describe the relationship between nouns. Thus, by spotting out the names of the nodes and/or attributes in a user's request, the system can determine which nodes and attributes will participate in the query topic and aspect.

124
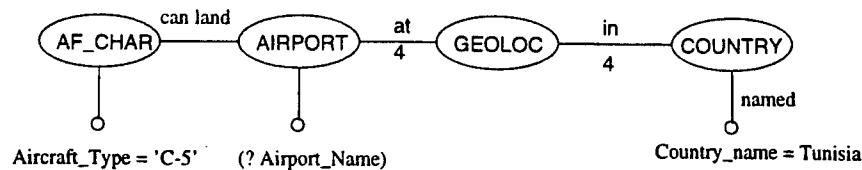
Figure 8: Augmented Query Topic for Natural Language Interface

As an example, consider the following request: "Find me an airport in Tunisia that can land a C-5." The request mentions *airport*, *Tunisia*, *can land*, and *C-5*. These keywords and phrases are mapped onto their corresponding components in the semantic graph. These components correspond to an incomplete query topic, which will be used by the high-level query formulator to formulate a query. Figure 8 shows the completed query topic which corresponds to the user's request. The query topic in the figure has been augmented with attribute nodes and attribute names.

The translation from a natural language request to a query topic is not so straightforward in general because of ambiguity. To help alleviate the ambiguity problem, the system takes advantage of constraints imposed by the database and by the semantic model. The query topic is used to narrow down the scope of possibilities. Using just constraints, however, often does not fully disambiguate a keyword, and statistical methods are used to make a best guess based on probabilities computed using past cases. A detailed description of the statistical method is beyond the scope of this paper.

Thus far, we have described how the query topic is obtained from the user's input, but we have not determined how the query aspect can be obtained. In the simple case, the attributes in the query aspect are explicitly mentioned in the user's request. For instance, "Find me the runway length of Bizerte airport", has the attribute *runway length* as its aspect, which is explicitly mentioned. More difficult cases include implicit aspect attributes and wh-questions. Implicit aspect attributes are those which are taken for granted by the user. An example of this is, "Find me an airport in Tunisia." In this request, the actual aspect attribute is not mentioned, but the system should know that the user is looking for a unique identifier for the node *airport*, which in our database, is the airport name. A wh-question begins with *who, what, where, why, when* or *how*. Usually, the user is seeking a specific aspect and the system must be able to determine the appropriate attribute based on the wh-word and the rest of the sentence. For instance, "Where is Bizerte airport?", should return a latitude and longitude pair. To do this, the system must

125

understand that *where* refers to a location. Since *airport* is the only node mentioned in the request, the system must conclude that the location of an airport is being sought.

4.4.2 Response Generation Using the Semantic Graph

After a query is processed by the CoBase system, responses are returned in the form of a set of tuples that meet the query constraints. This form of answer is good in a command-line system, but is insufficient when dealing with a voice interface. It is desirable to have responses in the form of English-like sentences, which could be displayed on the screen or could be read back using a voice synthesizer. The semantic graph can be used to generate such an English-like response.

English-like responses are generated from the semantic graph using the query topic and the query aspect, if there is one. Similar to generating a query description, we will make use of the node names, the attribute names and the link labels in the semantic graph to piece together an English-like response which reflects the user's original query and the answers obtained, if any.

When answers are returned by CoBase, an English-like response is generated based on the answers. Given a query aspect, we reiterate the user's query and add the descriptions of the answers at the end. For the request "Find me an airport in Tunisia that can land a C-5", the response is generated using the query topic in figure 8. If we traverse a path starting from the node *AIRPORT*, we get the following response: "Airport has airport name Bizerte at geographical location in country with country name Tunisia can land aircraft type C-5." Determining the ordering of the nodes and links in the response path is an open issue which requires more research.

4.4.3 Comparison to Other Systems

Our technique for translating the user's input in natural language to its corresponding database query is based on spotting keywords to map the input onto the semantic graph. This is similar to pattern matching systems [Wil75], which use templates to match certain patterns of words. The major disadvantage of pattern matching systems is that they produce very shallow analyses which may lead to very erroneous interpretations. Because our system maps objects onto the semantic graph, we can avoid shallow analyses because the meaning is captured and constrained by the semantics of the database. Other techniques used in natural language interfaces to databases include syntax-based and semantic grammars [ART95, GBG+96, YP89]. Both syntax-based and semantic grammars use context free grammars to produce parses which allow the system to interpret the user's input. The advantage of our keyword spotting technique is that we avoid
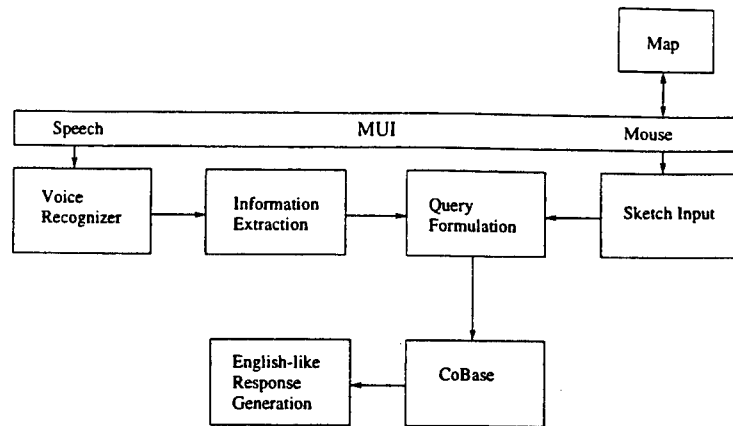
Figure 9: Query Formulation and Multimodal Interface System Architecture

using rules, which may not be able to scale to large domains, and our language limitation is mainly due to the terms used in the semantic graph. Because the semantic graph has the facilities to hold multiple names for nodes and attributes, the coverage of our domain-specific language may be sufficiently high enough for practical use.

5   Implementation and Experience

Figure 9 shows the architecture of our query formulation system with a multimodal user interface. For the voice input, we use IBM's ViaVoice, a commerical off-the-shelf voice recognizer which provides a API through which we can obtain useful information such as alternative guesses. The voice recognizer feeds into the language understanding component which uses the semantic graph to piece together a query topic and aspect, which is then fed into the query formulator. The user is also able to use a point-and-click device to designate a region in the map interface. This information is also used in the query formulation process. The formulated query is given to CoBase for processing and the answers returned are fed into the response generator. Using the semantic graph, the response generator produces an English-like response, which is then read back to the user using IBM's Virtual Voices. Since voice recognizers still tend to make many errors, the point-and-click interface is made available to the user to be used in conjunction with the voice system. For instance, if the voice system detects that it cannot understand what the user is saying, it may direct the user to use the point-and-click interface.

The query formulator was implemented in Java with three packages for the graph model, the query representation, and the formulator, with a total of about 50 classes and 7,000 lines

of code. The formulator was tested on both the Windows NT and Solaris platform using an Oracle database. The query formulator is JDBC compliant, therefore it can be ported to multiple platforms and databases.

The query formulator was tested on a transportation database provided by DARPA. There are about 300 relations and the largest relation has more than 50,000 tuples. The semantic graph was automatically generated from the database schema and a domain expert labelled some of the links with high-level concepts (e.g. 'CAN LAND'). Queries that involve up to 7 relations and contain multiple query contraints have been successfully formulated by the query formulator. CoSQL queries that involve coopertive operators like SIMILAR-TO, APPROXIMATE, RELAX-ORDER and NOT-RELAXABLE were also correctly formulated. To test its extensibility and portability, we also tested the query formulator on a different domain, a logistics database. The only effort required to port the query formulator to the logistics domain is the regeneration of the semantic graph and the relabelling of the links by a domain expert.

Performance tests were conducted on the transportation database schema for accuracy and search time on a Sun Ultrasparc II machine. The graph for the transportation database contains 50 nodes and 105 links. For 30 random query samples, all the first query candidate are correct for tree queries. The formulation search time ranges from 84 milliseconds to 871 milliseconds, with an average time of 323 milliseconds, which is well below the query execution time.

The query formulator provides a set of APIs that allow other applications to easily interface with it. We have integrated the query formulator with a point-and-click user interface, a map interface and a voice input interface.

6 Conclusion

A new approach for query formulation which is based on a semantic graph model is presented. The query formulation can be viewed as a graph search problem. The search goal is to find links and nodes from the semantic graph to complete the query and form a conneccted subgraph from the user input. When multiple queries are formulated from a user input, the user can resolve such query ambiguity based on ranking and English-like query descriptions. The query candidates can be ranked based on the information of nodes and links in the subgraph. The limit the search scope in finding the subgraph from the semantic graph, a heuristic algorithm was proposed to reduce the search complexities. The query formulator algorithm can formulate

the SELECT-PROJECT-JOIN queries with aggregate functions as well as CoSQL queries. We have constructed a prototype system using the above technique with point-and-click and voice interfaces. The system is currently operating on top of a cooperative database (CoBase) at UCLA to formulate SQL queries.

References

[ART95]   I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interface to databases - an introduction. *Journal of Natural Language Engineering*, 1:29–81, 1995.

[CBS94]   Roger H. L. Chiang, Terence M. Barron, and Veda C. Storey. Reverse engineering of relational databases: Extraction of an eer model from a relational database. *Data & Knowledge Engineering*, 12:107–142, 1994.

[Che83]   Peter P. Chen. English sentence structure and entity-relationship diagrams. *Information Sciences*, 29:127–149, 1983.

[CMB93]   Wesley W. Chu, M. A. Merzbacher, and L. Berkovich. The design and implementation of CoBase. In *Proceedings of ACM SIGMOD 93*, pages 517–522, Washington D. C., USA, May 1993.

[Cod88]   E. F. Codd. 'universal' relation fails to replace relational model. *IEEE Software*, page 4, June 1988.

[CYC+96] Wesley W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. Cobase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 1996.

[GBG+96] David Goddeau, Eric Brill, James Glass, Christine Pao, Michael Phillips, Joseph Polifroni, Stephanie Seneff, and Victor Zue. Galaxy: A human-language interface to on-line travel information. In *ICSLP '96*, 1996.

[HR92]   F. K. Hwang and Dana S. Richards. Steiner tree problems. *Networks*, 22:55–89, 1992.

[IL94]   Yannis E. Ioannidis and Yezdi Lashkari. Incomplete path expressions and their disambiguation. In *Proceedings of SIGMOD'94*, pages 138–149, 1994.

[Qui93]   J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, Sam Mateo, CA, 1993.

[TYF86]   T. J. Teorey, D. Yang, and J. P. Fry. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, 18(2):197–222, June 1986.

[Ull88]   Jeffrey D. Ullman. *Principles of database and knowledge-base systems, Vol.II*. Computer Science Press, 1988.

[Var88]   Moshe Y. Vardi. The universal-relation data model for logical independence. *IEEE Software*, pages 80–85, March 1988.

[Wil75]   Y. Wilks. An intelligent analyzer and understander of english. *Communications of the ACM*, 18(5):264–274, 1975.

[WS84]   Joseph A. Wald and Paul G. Sorenson. Resolving the query inference problem using steiner trees. *ACM Transactions on Database Systems*, 9(3):348–368, September 1984.

[YP89]   S .J. Young and C.E. Proctor. The design and implementation of dialogue control in voice operated database inquiry systems. *Computer Speech and Language*, 3:329–353, 1989.

DISTRIBUTION LIST

addresses	number of copies
CRAIG S. ANKEN AFRL/IFTB 525 BROOKS ROAD ROME, NY 13441-4505	5
UNIV OF CALIFORNIA, LOS ANGELES 1400 PETER UBERROTH BUILDING LOS ANGELES, CA 90024-1406	1
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ATTN: NAN PFRIMMER IIT RESEARCH INSTITUTE 201 MILL ST. ROME, NY 13440	1
AFIT ACADEMIC LIBRARY AFIT/LDR, 2950 P.STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1
AFRL/HESC-TDC 2698 G STREET, BLDG 190 WRIGHT-PATTERSON AFB OH 45433-7604	1

```
ATTN:  SMDC IM PL                                    1
US ARMY SPACE & MISSILE DEF CMD
P.O. BOX 1500
HUNTSVILLE AL 35807-3801


COMMANDER, CODE 4TL000D                              1
TECHNICAL LIBRARY, NAWC-WD
1 ADMINISTRATION CIRCLE
CHINA LAKE  CA  93555-6100


CDR, US ARMY AVIATION & MISSILE CMD                  2
REDSTONE SCIENTIFIC INFORMATION CTR
ATTN: AMSAM-RD-OB-R, (DOCUMENTS)
REDSTONE ARSENAL AL 35898-5000


REPORT LIBRARY                                       1
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545


ATTN:  D'BORAH HART                                  1
AVIATION BRANCH SVC 122.10
FOB10A, RM 931
800 INDEPENDENCE AVE, SW
WASHINGTON DC   20591

AFIWC/MSY                                            1
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016


ATTN:  KAROLA M. YOURISON                            1
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE
PITTSBURGH PA 15213


USAF/AIR FORCE RESEARCH LABORATORY                   1
AFRL/VSOSA(LIBRARY-BLDG 1103)
5 WRIGHT DRIVE
HANSCOM AFB  MA   01731-3004


ATTN:  EILEEN LADUKE/D460                            1
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730
```

```
OUSD(P)/DTSA/DUTD                              1
ATTN:  PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202


SOFTWARE ENGR'G INST TECH LIBRARY              1
ATTN:  MR DENNIS SMITH
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213-3890


USC-ISI                                        1
ATTN:  DR ROBERT M. BALZER
4676 ADMIRALTY WAY
MARINA DEL REY CA 90292-6695


KESTREL INSTITUTE                              1
ATTN:  DR CORDELL GREEN
1801 PAGE MILL ROAD
PALO ALTO CA 94304


ROCHESTER INSTITUTE OF TECHNOLOGY              1
ATTN:  PROF J. A. LASKY
1 LOMB MEMORIAL DRIVE
P.O. BOX 9887
ROCHESTER NY 14613-5700

AFIT/ENG                                       1
ATTN:TOM HARTRUM
WPAFB OH 45433-6583


THE MITRE CORPORATION                          1
ATTN:  MR EDWARD H. BENSLEY
BURLINGTON RD/MAIL STOP A350
BEDFORD MA 01730


ANDREW A. CHIEN                                1
SAIC CHAIR PROF (SCI APL INT CORP)
USCD/CSE-AP&M 4808
9500 GILMAN DRIVE, DEPT. 0114
LAJOLLA CA 92093-0114

HONEYWELL, INC.                                1
ATTN:  MR BERT HARRIS
FEDERAL SYSTEMS
7900 WESTPARK DRIVE
MCLEAN VA 22102
```

```
SOFTWARE ENGINEERING INSTITUTE                    1
ATTN:  MR WILLIAM E. HEFLEY
CARNEGIE-MELLON UNIVERSITY
SEI 2218
PITTSBURGH PA 15213-38990


UNIVERSITY OF SOUTHERN CALIFORNIA                 1
ATTN:  DR. YIGAL ARENS
INFORMATION SCIENCES INSTITUTE
4676 ADMIRALTY WAY/SUITE 1001
MARINA DEL REY CA 90292-6695


COLUMBIA UNIV/DEPT COMPUTER SCIENCE               1
ATTN:  DR GAIL E. KAISER
450 COMPUTER SCIENCE BLDG
500 WEST 120TH STREET
NEW YORK NY 10027


AFIT/ENG                                          1
ATTN:  DR GARY B. LAMONT
SCHOOL OF ENGINEERING
DEPT ELECTRICAL & COMPUTER ENGRG
WPAFB OH 45433-6583


NSA/OFC OF RESEARCH                               1
ATTN:  MS MARY ANNE OVERMAN
9800 SAVAGE ROAD
FT GEORGE G. MEADE MD 20755-6000



AT&T BELL LABORATORIES                            1
ATTN:  MR PETER G. SELFRIDGE
ROOM 3C-441
600 MOUNTAIN AVE
MURRAY HILL NJ 07974


ODYSSEY RESEARCH ASSOCIATES, INC.                 1
ATTN:  MS MAUREEN STILLMAN
301A HARRIS B. DATES DRIVE
ITHACA NY 14850-1313



TEXAS INSTRUMENTS INCORPORATED                    1
ATTN:  DR DAVID L. WELLS
P.O. BOX 655474, MS 238
DALLAS TX 75265



KESTREL DEVELOPMENT CORPORATION                   1
ATTN:  DR RICHARD JULLIG
3260 HILLVIEW AVENUE
PALO ALTO CA 94304
```

```
DARPA/ITO                                              1
ATTN:  DR KIRSTIE BELLMAN
3701 N FAIRFAX DRIVE
ARLINGTON VA 22203-1714


NASA/JOHNSON SPACE CENTER                              1
ATTN:  CHRIS CULBERT
MAIL CODE PT4
HOUSTON TX 77058


STERLING IMD INC.                                      1
KSC OPERATIONS
ATTN:  MARK MAGINN
BEECHES TECHNICAL CAMPUS/RT 26 N.
ROME NY 13440

HUGHES SPACE & COMMUNICATIONS                          1
ATTN:  GERRY BARKSDALE
P. O. BOX 92919
BLDG R11 MS M352
LOS ANGELES, CA 90009-2919

SCHLUMBERGER LABORATORY FOR                            1
  COMPUTER SCIENCE
ATTN:  DR. GUILLERMO ARANGO
8311 NORTH FM620
AUSTIN, TX 78720

DECISION SYSTEMS DEPARTMENT                            1
ATTN:  PROF WALT SCACCHI
SCHOOL OF BUSINESS
UNIVERSITY OF SOUTHERN CALIFORNIA
LOS ANGELES, CA 90089-1421

SOUTHWEST RESEARCH INSTITUTE                           1
ATTN:  BRUCE REYNOLDS
6220 CULEBRA ROAD
SAN ANTONIO, TX 78228-0510


NATIONAL INSTITUTE OF STANDARDS                        1
  AND TECHNOLOGY
ATTN:  CHRIS DABROWSKI
ROOM A266, BLDG 225
GAITHSBURG MD 20899

EXPERT SYSTEMS LABORATORY                              1
ATTN:  STEVEN H. SCHWARTZ
NYNEX SCIENCE & TECHNOLOGY
500 WESTCHESTER AVENUE
WHITE PLAINS NY 20604
```

```
NAVAL TRAINING SYSTEMS CENTER                    1
ATTN:  ROBERT BREAUX/CODE 252
12350 RESEARCH PARKWAY
ORLANDO FL 32826-3224


DR JOHN SALASIN                                  1
DARPA/ITO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714


DR BARRY BOEHM                                   1
DIR, USC CENTER FOR SW ENGINEERING
COMPUTER SCIENCE DEPT
UNIV OF SOUTHERN CALIFORNIA
LOS ANGELES CA 90089-0781

DR STEVE CROSS                                   1
CARNEGIE MELLON UNIVERSITY
SCHOOL OF COMPUTER SCIENCE
PITTSBURGH PA 15213-3891


DR MARK MAYBURY                                  1
MITRE CORPORATION
ADVANCED INFO SYS TECH: G041
BURLINTON ROAD, M/S K-329
BEDFORD MA 01730

ISX                                              1
ATTN:  MR. SCOTT FOUSE
4353 PARK TERRACE DRIVE
WESTLAKE VILLAGE,CA 91361


MR GARY EDWARDS                                  1
ISX
433 PARK TERRACE DRIVE
WESTLAKE VILLAGE CA 91361


DR ED WALKER                                     1
BBN SYSTEMS & TECH CORPORATION
10 MOULTON STREET
CAMBRIDGE MA 02238


LEE ERMAN                                        1
CIMFLEX TEKNOWLEDGE
1810 EMBACADERO ROAD
P.O. BOX 10119
PALO ALTO CA 94303
```

```
DR. DAVE GUNNING                                    1
DARPA/ISO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA   22203-1714


DAN WELD                                            1
UNIVERSITY OF WASHINGTON
DEPART OF COMPUTER SCIENCE & ENGIN
BOX 352350
SEATTLE, WA 98195-2350

STEPHEN SODERLAND                                   1
UNIVERSITY OF WASHINGTON
DEPT OF COMPUTER SCIENCE & ENGIN
BOX 352350
SEATTLE, WA 98195-2350

DR. MICHAEL PITTARELLI                              1
COMPUTER SCIENCE DEPART
SUNY INST OF TECH AT UTICA/ROME
P.O. BOX 3050
UTICA, NY 13504-3050

CAPRARO TECHNOLOGIES, INC                           1
ATTN:  GERARD CAPRARO
311 TURNER ST.
UTICA, NY 13501


USC/ISI                                             1
ATTN:  BOB MCGREGOR
4676 ADMIRALTY WAY
MARINA DEL REY, CA 90292


SRI INTERNATIONAL                                   1
ATTN:  ENRIQUE RUSPINI
333 RAVENSWOOD AVE
MENLO PARK, CA 94025


DARTMOUTH COLLEGE                                   1
ATTN:  DANIELA RUS
DEPT OF COMPUTER SCIENCE
11 ROPE FERRY ROAD
HANOVER, NH 03755-3510

UNIVERSITY OF FLORIDA                               1
ATTN:  ERIC HANSON
CISE DEPT 456 CSE
GAINESVILLE, FL 32611-6120
```

```
CARNEGIE MELLON UNIVERSITY                  1
ATTN:  TOM MITCHELL
COMPUTER SCIENCE DEPARTMENT
PITTSBURGH, PA 15213-3890


CARNEGIE MELLON UNIVERSITY                  1
ATTN:  MARK CRAVEN
COMPUTER SCIENCE DEPARTMENT
PITTSBURGH, PA 15213-3890


UNIVERSITY OF ROCHESTER                     1
ATTN:  JAMES ALLEN
DEPARTMENT OF COMPUTER SCIENCE
ROCHESTER, NY 14627


TEXTWISE, LLC                               1
ATTN:  LIZ LIDDY
2-121 CENTER FOR SCIENCE & TECH
SYRACUSE, NY 13244


WRIGHT STATE UNIVERSITY                     1
ATTN:  DR. BRUCE BERRA
DEPART OF COMPUTER SCIENCE & ENGIN
DAYTON, OHIO 45435-0001


UNIVERSITY OF FLORIDA                       1
ATTN:  SHARMA CHAKRAVARTHY
COMPUTER & INFOR SCIENCE DEPART
GAINESVILLE, FL 32622-6125


KESTREL INSTITUTE                           1
ATTN:  DAVID ESPINOSA
3260 HILLVIEW AVENUE
PALO ALTO, CA 94304


USC/INFORMATION SCIENCE INSTITUTE           1
ATTN:  DR. CARL KESSELMAN
11474 ADMIRALTY WAY, SUITE 1001
MARINA DEL REY, CA 90292


MASSACHUSETTS INSTITUTE OF TECH             1
ATTN:  DR. MICHAELE SIEGEL
SLOAN SCHOOL
77 MASSACHUSETTS AVENUE
CAMBRIDGE, MA 02139
```

```
USC/INFORMATION SCIENCE INSTITUTE                    1
ATTN:  DR. WILLIAM SWARTHOUT
11474 ADMIRALTY WAY, SUITE 1001
MARINA DEL REY, CA 90292


STANFORD UNIVERSITY                                  1
ATTN:  DR. GIO WIEDERHOLD
857 SIERRA STREET
STANFORD
SANTA CLARA COUNTY, CA 94305-4125

NCCOSC RDTE DIV D44208                               1
ATTN:  LEAH WONG
53245 PATTERSON ROAD
SAN DIEGO, CA 92152-7151


SPAWAR SYSTEM CENTER                                 1
ATTN:  LES ANDERSON
271 CATALINA BLVD, CODE 413
SAN DIEGO CA 92151


GEORGE MASON UNIVERSITY                              1
ATTN:  SUSHIL JAJODIA
ISSE DEPT
FAIRFAX, VA 22030-4444


DIRNSA                                               1
ATTN:  MICHAEL R. WARE
DDD, NSA/CSS (R23)
FT. GEORGE G. MEADE MD 20755-6000


DR. JIM RICHARDSON                                   1
3660 TECHNOLOGY DRIVE
MINNEAPOLIS, MN 55418



LOUISIANA STATE UNIVERSITY                           1
COMPUTER SCIENCE DEPT
ATTN:  DR. PETER CHEN
257 COATES HALL
BATON ROUGE, LA 70803

INSTITUTE OF TECH DEPT OF COMP SCI                   1
ATTN:  DR. JAIDEEP SRIVASTAVA
4-192 EE/CS
200 UNION ST SE
MINNEAPOLIS, MN 55455
```

```
GTE/BBN                                         1
ATTN:  MAURICE M. MCNEIL
9655 GRANITE RIDGE DRIVE
SUITE 245
SAN DIEGO, CA 92123

UNIVERSITY OF FLORIDA                           1
ATTN:  DR. SHARMA CHAKRAVARTHY
E470 CSE BUILDING
GAINESVILLE, FL 32611-6125


AFRL/IFT                                        1
525 BROOKS ROAD
ROME, NY 13441-4505


AFRL/IFTM                                       1
525 BROOKS ROAD
ROME, NY 13441-4505
```

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of information systems science and

technology for aerospace command and control and its transition to air,

space, and ground systems to meet customer needs in the areas of Global

Awareness, Dynamic Planning and Execution, and Global Information

Exchange is the focus of this AFRL organization. The directorate's areas

of investigation include a broad spectrum of information and fusion,

communication, collaborative environment and modeling and simulation,

defensive information warfare, and intelligent information systems

technologies.